AFRL-RY-WP-TM-2008-1205

CHARACTERIZATION OF PERIODICALLY POLED NONLINEAR MATERIALS USING DIGITAL IMAGE PROCESSING

James R. Alverson

Electro-Optical Countermeasures Technology Branch
Electro-Optical Sensor Technology Division

APRIL 2008
Interim Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

*//Signature//

KENNETH L. SCHEPLER, Principal Scientist
EO CM Tech Branch, EO Sensor Tech Div
Sensors Directorate

//Signature//

JOHN F. CARR, Chief
EO CM Tech Branch, EO Sensor Tech Div
Sensors Directorate

//Signature//

BRIAN C. FORD, Col, USAF
Chief, EO Sensor Technology Division
Sensors Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YY)* April 2008	**2. REPORT TYPE** Interim	**3. DATES COVERED *(From - To)*** 01 April 2007 – 22 April 2008

4. TITLE AND SUBTITLE CHARACTERIZATION OF PERIODICALLY POLED NONLINEAR MATERIALS USING DIGITAL IMAGE PROCESSING	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER In-house
	5c. PROGRAM ELEMENT NUMBER 61102F/62204F
6. AUTHOR(S) James R. Alverson	**5d. PROJECT NUMBER** 2003
	5e. TASK NUMBER 12
	5f. WORK UNIT NUMBER 20031224
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Electro-Optical Countermeasures Technology Branch (AFRL/RYJW) Electro-Optical Sensor Technology Division Air Force Research Laboratory, Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command, United States Air Force	**8. PERFORMING ORGANIZATION REPORT NUMBER**

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command United States Air Force	Air Force Office of Scientific Research (AFOSR) 875 N. Randolph Street Arlington, VA 22203-1768	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/RYJW
		11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RY-WP-TM-2008-1205

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
This is the author's Master's Thesis through the University of Dayton's Electro-Optic program in the School of Engineering.

PAO Case Number: WPAFB 08-3540; Clearance Date: 27 Jun 2008. The paper contains color.

14. ABSTRACT
For as long as periodically poled devices have been produced, there has been a need to evaluate and improve their performance. A new approach based on image processing across an entire z+ or z- surface of a poled crystal allows for better quantification of the underlying domain structure and directly relates to device performance. The poled regions are determined from the images and this mapping is then used to determine an effective nonlinear d-coefficient. Experimentally validated measurements are shown for an optical parametric generator setup. Future improvements in the poling process should be possible by using this technique to produce quantifiable metrics to compare device fabrication iterations, which has applications in Quality Assurance as well as device development.

15. SUBJECT TERMS
Nonlinear Optics, Periodically Poled Materials, Ferroelectric Materials, Microscopy, Digital Image Processing

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor) Kenneth L. Schepler
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	56	**19b. TELEPHONE NUMBER** *(Include Area Code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Table of Contents

List of Illustrations

Acknowledgements

I would like to thank Dr. Ken Schepler from the EO Countermeasures Technology Branch of the Sensors Directorate at the Air Force Research Lab for his invaluable advice, answers, questions and overall guidance on this project. I would also like to thank my advisor Dr. Peter Powers from the University of Dayton for his help in both research and academic matters. Finally, I'd like to thank Sarah for helping me stay motivated so that I could make it to the end.

Chapter 1
Introduction

For as long as quasi-phased matched devices have been produced, there has been a need to evaluate and improve their performance. This evaluation has most often been done qualitatively by spot checking domain boundaries of etched surfaces using a light microscope or quantitatively by using the crystal in a power measurement to determine conversion efficiency. Spot checking, however, only gives coarse information on whether the device will be effective, and the conversion efficiency measurement requires that an experimental setup be available for the purpose of testing the crystal. This is not always practical, especially for crystal vendors who serve a wide variety of applications which would require a host of lasers to test the crystals. Furthermore, as quasi-phase matched designs become more complex, such as using chirped structures[1] and even aperiodic structures[2], an evaluation of how close the poled structure reproduces the mask design is required.

A new approach to better quantify the underlying domain structure was developed to determine the device performance. Previously, the best way to quantitatively compare two periodically poled crystals without testing with a laser was by measuring numerical statistics on the domain period.[3] Typically, this would be from a representative region of the crystal, instead of using the complete domain structure. In this thesis, the use of image processing across an entire z+ or z- surface of a poled crystal is explored to characterize the domain structure and then is compared to the results of a representative region. The poled regions are mapped by finding regions of maximum curvature in the image surface, or alternatively maximum derivatives, corresponding to the domain boundary edges produced during the etching process, and the mapping is used to determine an effective nonlinear coefficient. Experimentally validated measurements from an optical parametric generation (OPG) setup are consistent with the calculated nonlinear coefficient across the crystal. This technique would be useful for evaluating the quality of device iterations for either research or production.

1.1 Introduction to Nonlinear Optics

Nonlinear optical crystals have many useful purposes. One of the most common applications is converting the wavelength of a laser to one or more other wavelengths. Indirectly generating wavelengths of light at atmospheric transmission windows, for example, is useful for military applications, such as laser radar, infrared countermeasures, remote sensing, and target recognition and designation. Commercial applications include spectroscopy, displays, and various medical technologies.

Nonlinear optical interactions in these crystals work by driving a nonlinear polarization field, P, often but not necessarily expressed as a power series in the field strength, E, shown in Eq. (1) as

$$\begin{aligned}\tilde{P}(t) &= \chi^{(1)}\tilde{E}(t) + \chi^{(2)}\tilde{E}^{2}(t) + \chi^{(3)}\tilde{E}^{3}(t) + \ldots \quad (1)\\ &\equiv \tilde{P}^{(1)}(t) + \tilde{P}^{(2)}(t) + \tilde{P}^{(3)}(t) + \ldots\end{aligned}$$

where χ is the susceptibility, the tilde denotes a quantity that varies rapidly in time, and the superscript (n) denotes the n-th order. The first order susceptibility is also known as the linear susceptibility and is related to the index of refraction and dispersion. The focus of this research deals with $\chi^{(2)}$ processes.

The general idea of nonlinear optical processes in materials is that the incident field strength, E, induces oscillating dipole moments in each of the atoms. A thorough description of the origin and physics of nonlinear optical processes is given by Boyd.[4] Due to the nonlinear nature of the response, a nonlinear polarization at new frequencies is generated which can radiate at frequencies not present in the incident radiation field. This coupling allows energy to be transferred between different wavelengths and forms the basis of the physical mechanism behind these processes. An isolated atom would radiate in the typical dipole radiation pattern, but in a material, a large number of dipoles are oscillating with a phase related to the incident fields producing a phased array of dipoles. With nonlinear interactions, the incident fields set up a nonlinear polarization which needs to be phased up with the freely propagating field. If the dipoles radiate in phase with each other, then the resulting radiation will add constructively and form a narrow beam. This process is somewhat analogous to a phased array of antenna elements that radiate at radio frequencies.

1.2 Nonlinear Optical Interactions

The following derivation for the propagation of plane wave light through a (lossless) nonlinear optical material is provided with extensive details by Boyd.[4] Beginning with Maxwell's equations, shown in Eq. (2) through Eq. (5),

$$\nabla \cdot \tilde{D} = 4\pi\tilde{\rho} \quad (2)$$

$$\nabla \cdot \tilde{B} = 0 \quad (3)$$

$$\nabla \times \tilde{E} = -\frac{1}{c}\frac{\partial \tilde{B}}{\partial t} \quad (4)$$

$$\nabla \times \tilde{H} = \frac{1}{c}\frac{\partial \tilde{D}}{\partial t} + \frac{4\pi}{c}\tilde{J}\,, \quad (5)$$

it is assumed there are no free charges (ρ = 0), no free currents (J = 0), and the material is nonmagnetic (B = H). The nonlinear nature of the material is reflected in the displacement field relationship shown in Eq. (6),

$$\tilde{D} = \tilde{E} + 4\pi\tilde{P}\,. \quad (6)$$

By taking the curl of Eq. (4) and substituting in Eq. (5), the equation eventually simplifies for the nonlinear processes discussed shortly into an inhomogeneous wave equation, shown in Eq. (7),

$$\nabla^2\tilde{E} - \frac{n^2}{c^2}\frac{\partial^2\tilde{E}}{\partial t^2} = \frac{4\pi}{c^2}\frac{\partial^2\tilde{P}^{NL}}{\partial t^2}\,, (7)$$

where n is the index of refraction and P^{NL} is the 2nd and higher order polarizations.

Consider an incident field made up of two independent frequency components producing a second order polarization through the nonlinear susceptibility. The incident field is shown in Eq. (8),

$$\tilde{E}(t) = E_1 e^{-i\omega_1 t} + E_2 e^{-i\omega_2 t} + c.c., \qquad (8)$$

which produces the polarization, shown in Eq. (9),

$$\tilde{P}(t) = \chi^{(2)}\left[E_1{}^2 e^{-2i\omega_1 t} + E_2{}^2 e^{-2i\omega_2 t} + 2E_1E_2 e^{-i(\omega_1+\omega_2)t} + 2E_1E_2{}^* e^{-i(\omega_1-\omega_2)t} + c.c.\right] \qquad (9)$$
$$+ 2\chi^{(2)}\left[E_1E_1^* + E_2E_2{}^*\right].$$

These terms represent the possible 2nd order nonlinear interactions, but typically only one term in Eq. (9) will be used since it is unlikely that more than one term will satisfy the phase matching condition simultaneously.

1.2.1 Difference Frequency Generation (DFG)

In one second order nonlinear interaction called DFG, shown in Figure 1, two optical waves at frequencies ω_p and ω_s, representing the pump and signal waves, interact in the nonlinear optical material to produce the idler wave at frequency $\omega_i = \omega_p - \omega_s$. Following the derivation in Boyd, ultimately leads to a set of coupled amplitude equations, shown in Eq. (10) and Eq. (11),

$$\frac{dA_s}{dz} = \frac{8\pi i \omega_s{}^2 d_{eff}}{k_s c^2} A_p A_i{}^* e^{i\Delta kz} \quad (10)$$

$$\frac{dA_i}{dz} = \frac{8\pi i \omega_i{}^2 d_{eff}}{k_i c^2} A_p A_s{}^* e^{i\Delta kz} \quad (11)$$

where

$$E_m = A_m e^{ik_m z}, \qquad \text{m = p, s, or i}$$

$$d_{eff} = \frac{1}{2}\chi^{(2)} \qquad k_m = \frac{n_m \omega_m}{c}$$

$$\Delta k = k_p - k_s - k_i$$

The coupled amplitude equation for the pump wave is not provided because the pump amplitude is assumed to be undepleted, or the relative spatial change is so small as to be negligible.

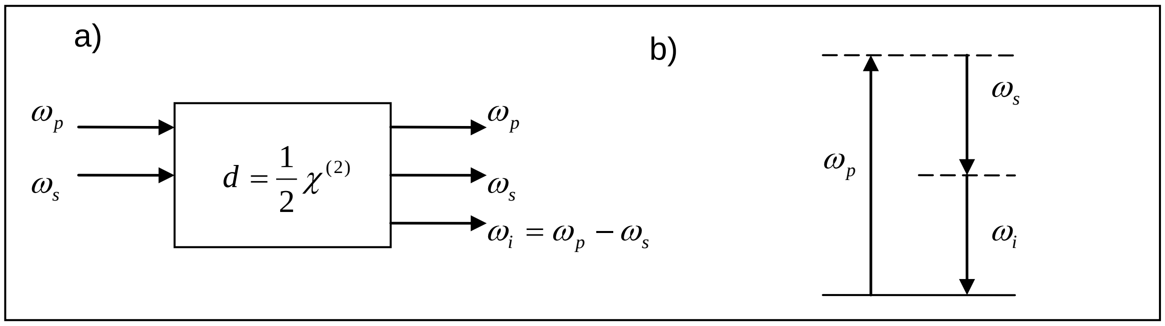

Figure 1. Difference frequency generation a) system and b) photon energy level diagram.

The phase-matched condition is $\Delta k = 0$ and any deviation reduces the efficiency. The reason is that the exponential in Eq. (11) containing Δk is an oscillating term in the z-direction. For DFG with two undepleted pumps, we can simply integrate this equation. The result is that when Δk is close to zero, the spatial extent of the oscillation is larger than the crystal length leading to a net positive after integration. The largest integration will occur when Δk = 0, since the exponential term is equal to unity. Once the oscillation's period is shorter than the crystal length, the oscillation tends to have nearly parts positive and negative, which counteract each other so the field does not get a chance to build up. The result is that for Δk away from zero, the integration equals nearly zero.

1.2.2 Optical Parametric Generation (OPG)

Optical parametric generation is a process where the signal and idler fields are spontaneously generated. It is sometimes also known as spontaneous parametric fluorescence. The pump photon still splits into a signal and an idler photon, but the signal and idler's wavelengths are determined by the phase matching condition and the energy conservation principle. The phenomenon can be described semi-classically as being started by quantum vacuum fluctuations that allow emissions to occur at any wavelength and in any direction which satisfy energy and momentum conservation. To account for this, the signal and idler can be initially set to ½ of a photon or the signal can be set to one photon and the idler set to none.

The OPG process does not have physical threshold like other optical processes since there will always be some output. Instead a "threshold" is determined by the lowest detectable output. The pump threshold, therefore, will be the pump power (or energy) that produces a detectable output (signal and idler). To calculate this threshold, the expected signal power collected by a detector subtending an angle 2θ for a given pump power assuming small gain is calculated from Eq. (12)[5],

$$P_s \approx \frac{\pi \beta L P_p}{|b|} \theta^2 \qquad (12)$$

where

$$\beta = \frac{\hbar \omega_i \omega_s^4 n_s d_{eff}^{\ 2}}{2\pi^2 \varepsilon_0 n_i n_p c^5} \text{ and } b = \left(\frac{dk_s}{d\omega_s}\right)_{\omega_s = \omega_{s0}} - \left(\frac{dk_i}{d\omega_i}\right)_{\omega_i = \omega_{i0}} .$$

In the equation, n is index of refraction, ω is angular frequency, k is wavevector, c is speed of light, ε_0 is the dielectric constant of free space, ħ is Dirac's constant, L is length, P is the power, the subscripts p, s, and i represent the pump, signal, and idler, respectively, and the subscript 0 refers to the frequency at perfect phase matching. By looking at the output of the OPG at the threshold, we ensure that the undepleted pump (small gain) approximation is valid. When translating laterally across a crystal using quasi-phase matching (defined below), all of the parameters will be constant except for d_{eff} (due to variations in the quality of the quasi-phase matching), so a scaled relative threshold is inversely proportional to d_{eff}^2.

1.3 Quasi-Phase Matched Analysis

There are two main ways to achieve phase matching: birefringent phase matching (BPM) and quasi-phase matching (QPM). BPM uses the birefringence of a crystal to compensate for the dispersion of the linear refractive indices. This typically involves orienting a crystal and laser beam to achieve the desired interaction. When BPM is not normally possible or more flexibility is desired (different output wavelengths), QPM can also create a desired phase matching condition. A review of recent advances in QPM is given by Hum and Fejer.[6] The essential idea is modulation of the nonlinear susceptibility as a function of propagation distance in a crystal. As described with birefringent phase matching above, if Δk is not equal to zero, the field will build up and then decay. The reason for this is that the nonlinear polarization at a given position in the crystal radiates and this radiation can interfere with the freely propagating field generated earlier in the crystal. Hence the field builds up for one coherence length and then decays for another coherence length. At the point where the field starts to decay, one can flip the nonlinear polarization to essentially reset the phasing so that the field continues to build up. For a sign inversion of the d coefficient, this results in a 180 degree phase shift of the nonlinear polarization. By spacing these inversions by one coherence length, the field will grow, although at a slower rate than perfect phase matching. The size, shape, and orientation of the designed crystal structure determine which output wavelengths satisfy the phase matching condition, unlike BPM, where optical material properties limit the possibilities. Also, larger nonlinear coefficients, unavailable for BPM, can be selected.

QPM materials can be produced in several ways. Originally devised by Armstrong et al.[7], early attempts at producing QPM materials involved stacking rotated crystals and diffusion bonding them together.[8] This idea was abandoned for much better results using lithography to periodically pole ferroelectrics with a large electric field.[9,10] The periodical poling of a crystal is the most common method for creating a QPM material. In periodic poling, an original bulk crystal with the z+ axis facing upward is transformed into a crystal with domains, or region with a uniform crystal orientation, that alternate between z+ axis being up and down, shown in Figure 2. The effect of this transformation is to produce domains that alternate between $+d_{eff}$ and $-d_{eff}$ across the crystal, effectively converting the d coefficient from a constant to a spatially varying pattern.

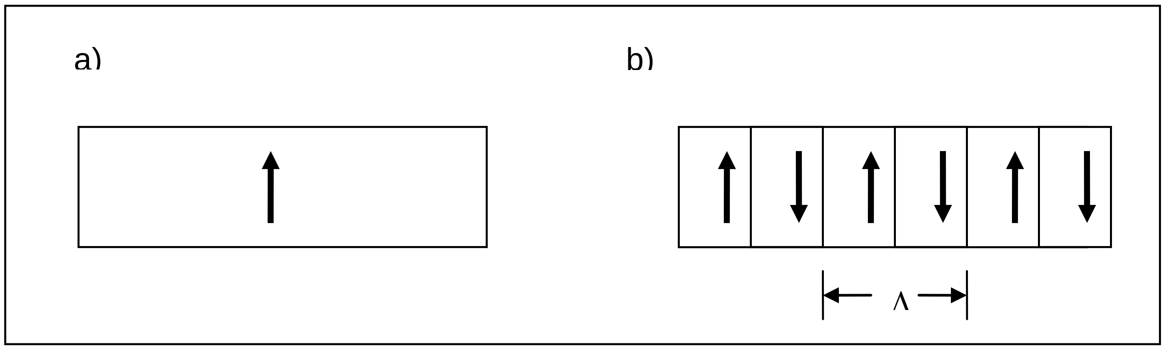

Figure 2. Representation of QPM through electric field periodic poling. A) Homogeneous single crystal (before poling) and b) a periodically poled material with alternating inverted crystal orientations and Λ spatial period.

The effect of a quasi-phase matched material on the DFG and OPG processes is to change the phase matching condition and the relative strength of the d coefficient. The first step is to include a spatially varying d coefficient, $d(z) = g(z)d_{eff}$, in the coupled amplitude equations, Eq. (10) and Eq. (11), to form Eq. (13) and Eq. (14),

$$\frac{dA_s}{dz} = \Gamma_s d_{eff} A_i^* g(z) e^{-i\Delta k' z} \qquad (13)$$

$$\frac{dA_i}{dz} = \Gamma_i d_{eff} A_s^* g(z) e^{-i\Delta k' z} \qquad (14)$$

where

$$\begin{aligned} \Gamma_m &= \frac{8\pi i \omega_m{}^2}{k_m c^2} A_p \\ \Delta k' = -\Delta k &= k_s + k_i - k_p \end{aligned} \quad , \ \mathrm{m = s, i} \ .$$

For DFG with a strong undepleted pump and signal (slowly varying amplitude approximation), the idler amplitude as it leaves the crystal simplifies to several constants multiplied by the Fourier transform of the g(z), as described by Fejer et al.[3] and as shown in Eq. (15),

$$\begin{aligned} A_i(L) &= \Gamma_i d_{eff} A_s^* \int_0^L g(z) e^{-i\Delta k' z} dz \\ &= \Gamma_i d_{eff} A_s^* \Im\{g(z)\} \end{aligned} \quad . \qquad (15)$$

The limits on the integral result from the finite extent of the nonlinear coefficient (within the crystal). The result from normal phase matching when g(z) = 1 within the crystal and g(z) = 0 everywhere else is shown in Eq. (16),

$$A_i(L) = ie^{-i\Delta k' L/2} \Gamma_i A_s^* d_{eff} \operatorname{sinc}(\Delta k' L/2). \qquad (16)$$

To solve for a spatially periodic nonlinear coefficient, g(z) is first written in terms of its Fourier series, shown in Eq. (17),

$$g(z) = \sum_n G_n e^{iK_n z} \ . \qquad (17)$$

where $K_n = 2\pi / \Lambda$ is the fundamental spatial frequency, and Λ is the spatial period. The ideal structure for periodic poling would be a square wave modulation from -1 to 1 with a 50% duty cycle. In this case, the $G_n = 2/(n\pi)$ and the new coupled amplitude equation looks like Eq. (18),

$$\begin{aligned} A_i(L) &= \Gamma_i d_{eff} A_s^* \int_0^L \sum_n G_n e^{iK_n z} e^{-i\Delta k' z} dz \\ &= \Gamma_i d_{eff} A_s^* \int_0^L \sum_n G_n e^{-i\Delta K z} dz \end{aligned} \quad . \qquad (18)$$

where $\Delta K = k_s + k_i - k_p - K_n$, indicating a new phase matching condition. Only one of the G_n coefficients will be close enough to ΔK = 0 to dominate the integral, so the series collapses to a single value based on the desired interaction. The final result is similar to Eq. (16), as seen in Eq. (19),

$$\begin{aligned} A_i(L) &= \Gamma_i d_{eff} G_n A_s^* \int_0^L e^{-i\Delta K z} dz \\ &= ie^{-i\Delta KL/2} \Gamma_i A_s^* d_n \sin c(\Delta KL/2) \end{aligned} \quad . \qquad (19)$$

where $d_n = d_{eff}G_n$.

A non-ideal periodic structure such as square wave with duty cycle variations would have a similar Fourier spectrum to the ideal square wave, except each ideal peak (corresponding to each G_n) would have a reduced magnitude and a broadened line. These can be calculated by changing the spatial period to the crystal length for the Fourier series calculation. If the variation in duty cycle is modest, the broadening would be small compared to the dominant peak and the phase matching bandwidth (sinc function) would still be small. Any nonlinear coefficient contribution from off the ΔK=0 peak would diminish quickly, allowing the main peak to remain dominant. Even with a non-ideal square wave, the amplitude of the main peak of a desired interaction will be used for measuring its nonlinear coefficient strength. For the same reasons, g(z) in the OPG coupled amplitude equations can be replaced with the value of the main peak near phase matching. For more complex patterns like chirped structures, g(z) would no longer be able to be reduced to a single number and a more sophisticated calculation would be required to determine the relative efficiency and bandwidth.

1.4 Visualization Techniques

The quantitative characterization of QPM nonlinear optical materials requires the use of a visualization technique before it can be digitized and processed to calculate performance metrics. A review of many of the visualization techniques for ferroelectric domains is given by Soergel.[11] Many of those methods of visualization were considered but either did not have enough resolution (better than 1μm) or the acquisition of an entire crystal would have taken too long. The options examined were using a SEM, a new technique involving doping Rb into KTP, and visual light microscopy of differentially etched Lithium niobate and KTP crystals.

1.4.1 Scanning Electron Microscope (SEM)

SEM imaging is one domain imaging technique that provides excellent resolution and is relatively easy and fast to perform for crystals such as lithium tantalate[12] and KTP[13]. Using a SEM with certain settings produces a domain contrast due to screening of charge which differs between the z+ surface and the z- surface due to heating from the e-beam and the pyroelectric effect as seen in Figure 3. With other settings, the domain walls become dark or bright due to a change in surface profile height from the piezoelectric effect and the orientation of the detector as seen in Figure 4. It is possible to image an entire crystal using either method.

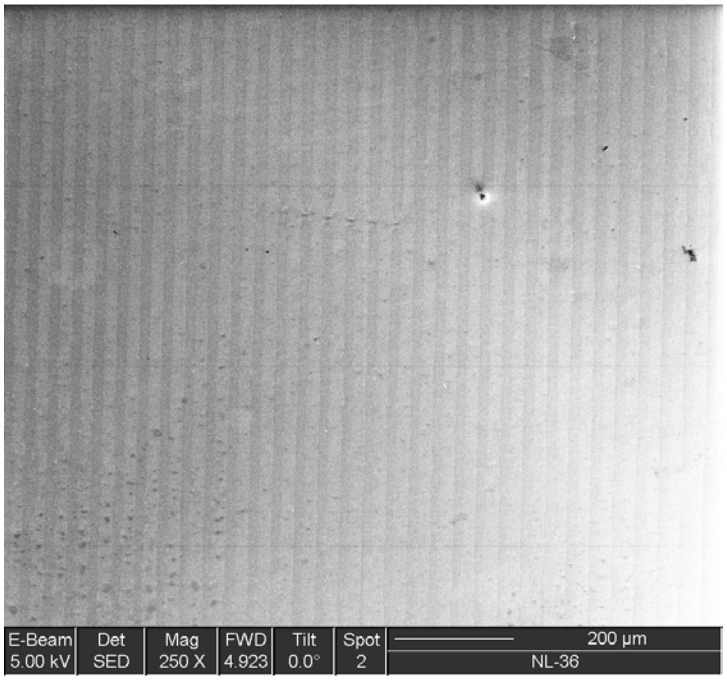

Figure 3. SEM image of periodically poled KTP showing intensity contrast.

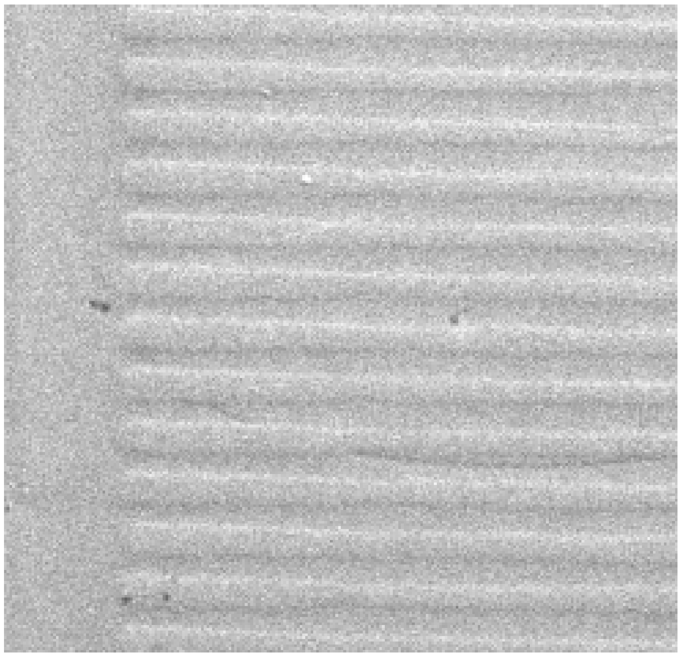

Figure 4. Close-up SEM image of periodically poled KTP showing boundary contrast.

1.4.2 Rb-doped KTP

A new technique for domain visualization was developed for KTP. The idea was to dope KTP with rubidium by dipping it into a $RbNO_3$ melt, which differentially dopes the surface (up to a micron in depth) depending on the domain (z+ or z-) which is exposed.[14] By taking a poled KTP sample, doping it, and then examining the surface for its level of Rb concentration, a map of the domain structure can be produced. The surface composition can be measured using a SEM incorporating an energy dispersive spectroscopy (EDS) system. This system performs x-ray microanalysis which can detect the various elements within a material. The results were not as impressive as hoped, however. Several immersion times were tested with different pieces of a commercially bought periodically poled KTP sample. A 100 nm layer of gold was sputtered on the crystal pieces before viewing under the SEM to avoid the accumulation of static electric charge during electron irradiation. The thickness of the gold was smaller than the depth of the electron beam's interaction volume within the sample, which can extend up to 5 μm, so all of the elements within the compound were still detectable. The resulting images in all cases were similar, with the best regular SEM vs. Rb image comparison shown in Figure 5. Not only did the Rb images have poor contrast and definition, but the image acquisition took much longer than anticipated (hours) for an image of a couple hundred microns on a side. This was due to the low detector count rate for the energy level corresponding to Rb.

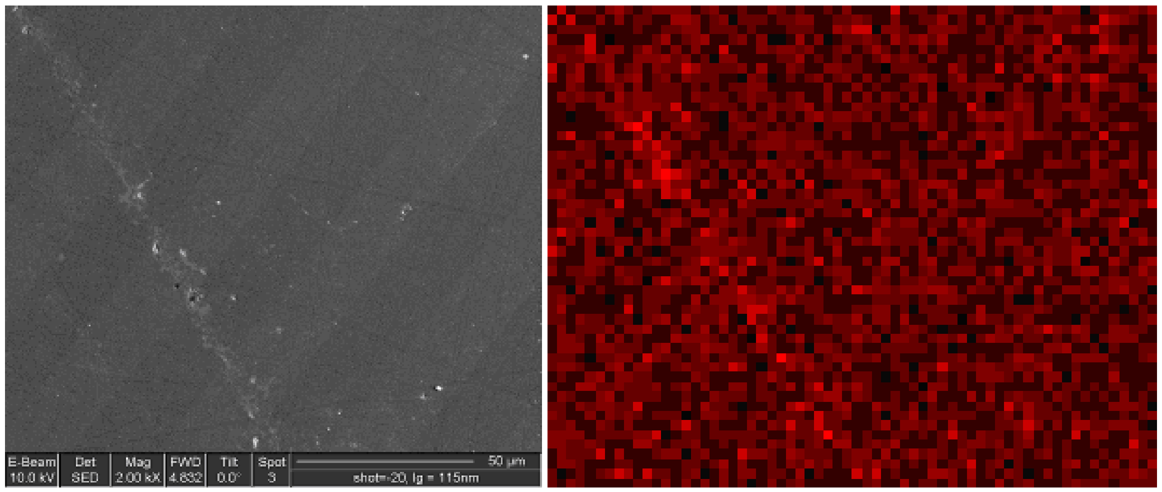

Figure 5. Side-by-side comparison of regular SEM and EDS image of Rb density.

1.4.3 Chemical Etch

The differential etching of periodically poled lithium niobate and KTP allows the domains of these crystals to be seen easily using a visible light microscope. The acid HF etches lithium niobate crystal surfaces at different rates for the z+ and z- surfaces.[15] The discontinuities in the surface profile produce dark edges under a microscope as seen in Figure 6. A mixture of KNO_3 and KOH in heated deionized water etches the surfaces of KTP at different rate for the z+ and z- surfaces.[16] A discontinuity is visible for etched KTP, also, as seen in Figure 7. Although these methods damage the surfaces of the crystals to some degree, the etching process is quick, the resolution is good, and taking pictures of an entire crystal under microscope is feasible. This technique was chosen for imaging the QPM domains.

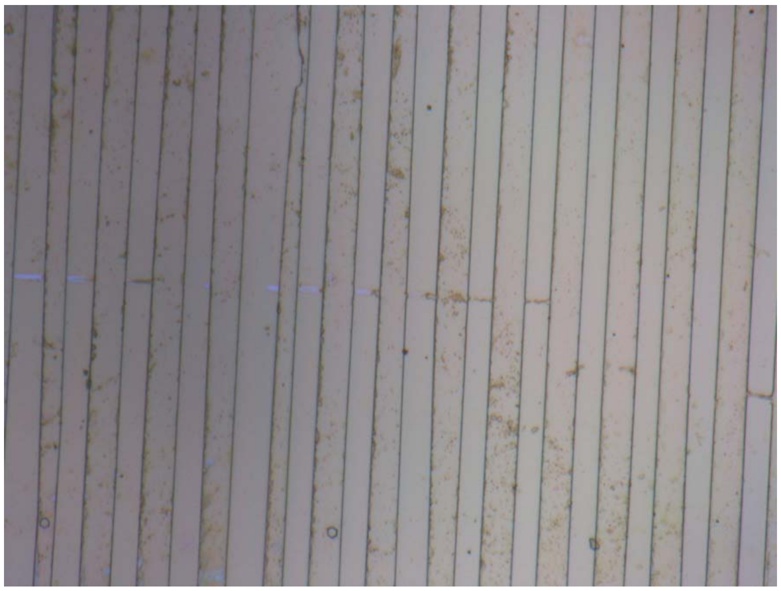

Figure 6. Picture of etched periodically poled lithium niobate under a visible light microscope.

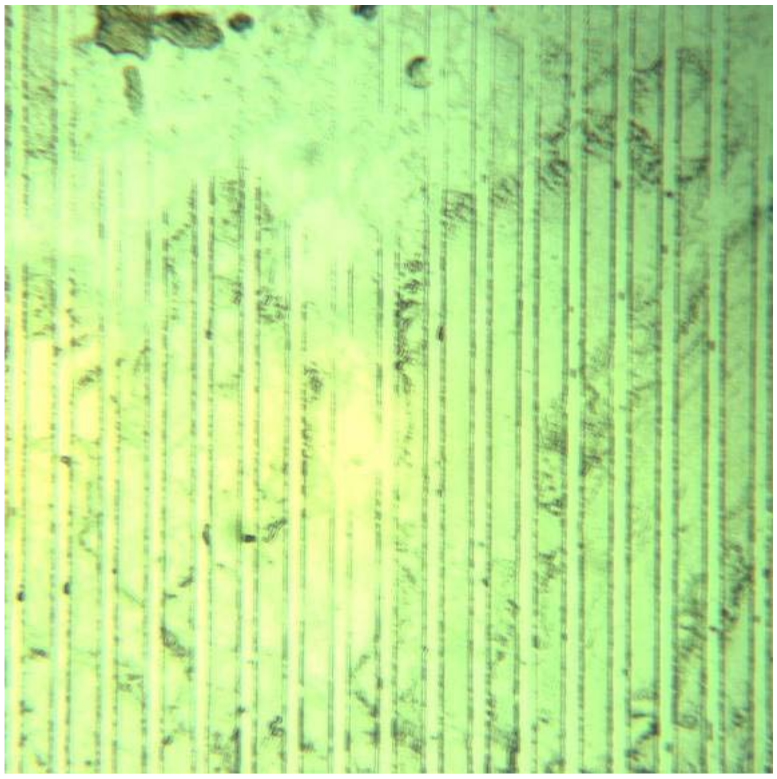

Figure 7. Picture of etched periodically poled KTP under a visible light microscope.

Chapter 2
Methodology

The main tasks for this experiment are (a) to obtain an image of the entire crystal, (b) to map the crystal domains by determining the positions of the domain boundaries through image processing, (c) to calculate the modified effective nonlinear coefficient, d_{eff}, and then (d) to compare it to measurements based on an OPG setup. The relative spatial variations in threshold and output (signal and idler) energy measurements should match the calculations based on the calculated d_{eff}.

2.1 Image Registration

To map the crystal domains of a periodically poled material, pictures of an entire z+ or z- face must be acquired. This can be accomplished by taking a series of slightly overlapping pictures (using a motorized stage or manually) of an etched material, in this case periodically poled lithium niobate (PPLN), using a reasonably high magnification optical microscope (Nikon AZ-100). Some software packages will combine this into a single image resulting in a single massive file, but such a large file can be difficult to use. An alternative is to calculate how to align the pictures by registering them together. There are many strategies for registration[17], but the correct technique often depends on the circumstances. Since poled materials are often periodic, the use of cross-correlation alone can be complicated due to the many peaks that would result.

A feature-based registration technique is more appropriate for our application since poling imperfections and dirt in the overlapping regions can be used as features for aligning. The features can be extracted manually or by looking for locations where there is a maximum curvature or derivative along both coordinate axes to find a sharp corner in the domain boundary or the location of a speck of dust. Since the images are obtained using a raster pattern with a consistent amount of overlap, the translation relationship between the features can be estimated. The estimate will be more accurate if the images are collected using a motorized stage with a specified step size due to its accuracy and precision. In this initial investigation, manual registration was used before any image processing was performed.

Before the images can be combined, any distortion or aberrations should be corrected, if necessary, to avoid an incorrect registration. While most aberrations lead to an image that appears fuzzy or out of focus, our images were clear and in focus. It is possible to see a small amount of lateral chromatic aberration, but this can be avoided by only using the green color component and calibrating the vertical and horizontal scales accordingly. Scale of the images was not important for the measurements in this experiment, so it was not determined. Barrel or pincushion distortion would likely be the only other significant issue, but it was not noticeable in the many nearly straight lines of the domains. Also, the amount of this distortion was calculated using a calibration slide, which has many evenly spaced, straight lines. The amount of distortion (percent difference between actual and expected spacing of lines) did not exceed 0.5%, which was believed to be a negligible factor at this point and therefore was not corrected. The distortion would be more of a concern when the wavelengths involved are important, which was not the case in this experiment. Distortion will be examined more closely as the development of image processing techniques progresses.

By simply scanning the microscope across the crystal, the transformation to relate the positions in one image to another should only involve translation. There would not be any scaling and the rotation should be negligible. However, this will depend on the translation stage, and care should be taken to make sure the stage has minimal pitch or yaw variations. After estimating the adjacent images position, a cross-correlation using a small window and centered at each feature was performed against its corresponding location in the other image. The resulting peak provided the relative location from the initial guess. The translation was determined for all images relative to the first image. The registration could be further refined by using the domain boundaries found later as features to match, but this was not done in this present work.

2.2 Image Processing

Using a representative microscope image of the etched lithium niobate, shown in Figure 8, the image processing techniques are described in detail below. A filtering technique using a sampled radial Gaussian function, G, shown in Eq. (20),

$$G(x,y,t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/2t}, \tag{20}$$

where x and y are integers and t is a thickness parameter, helps to smooth out noise without adding any artifacts to the image. By convolving this function with the image, I, shown in Eq. (21),

$$L(x,y,t) = G(x,y,t) * I(x,y), \tag{21}$$

the result, known as a scale-space representation[18] of the image, L, is obtained. A scale-space representation is useful as a preprocessing step because it smoothes features smaller than the scale t, so that only details of a certain scale are seen. Usually, multiple scales of the scale-space representation are used to differentiate features by their relative size in images. In our case, since the lines are a consistent size, we will only use one of the scale-space representations, although multiple scales could be used in the future to exclude large features from consideration or establish the crystal’s outer edges.

The visual effect of this type of filter is seen in Figure 9. The effect of the Gaussian filter is also seen in line scans shown in Figure 10. The larger Gaussian filter blocks are much slower due to the convolution operation. An alternative to the convolution is to Fourier transform the image and multiply by the Fourier transform of the Gaussian function, which is another Gaussian function. For this work, the 7x7 block filter (t = 2) was used because it reduces much of the noise while retaining the overall shape and amplitude of the peaks.

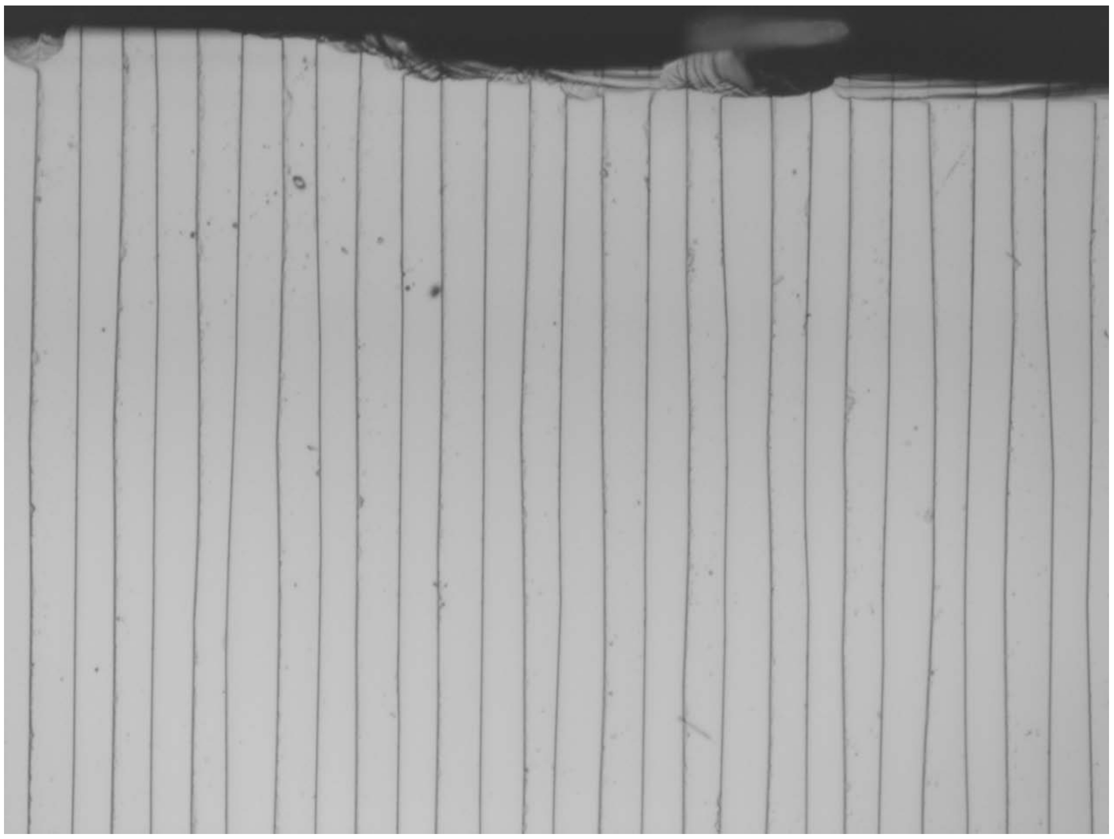

Figure 8. Etched lithium niobate test sample image.

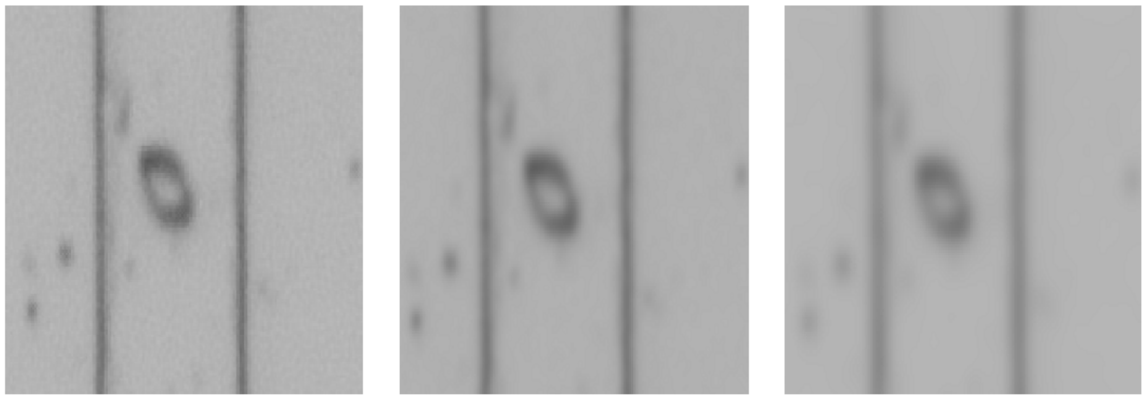

Figure 9. Section of original image (left), after t = 2, 7x7 pixel Gaussian filter (middle), and after t = 10, 61x61 pixel Gaussian filter (right).

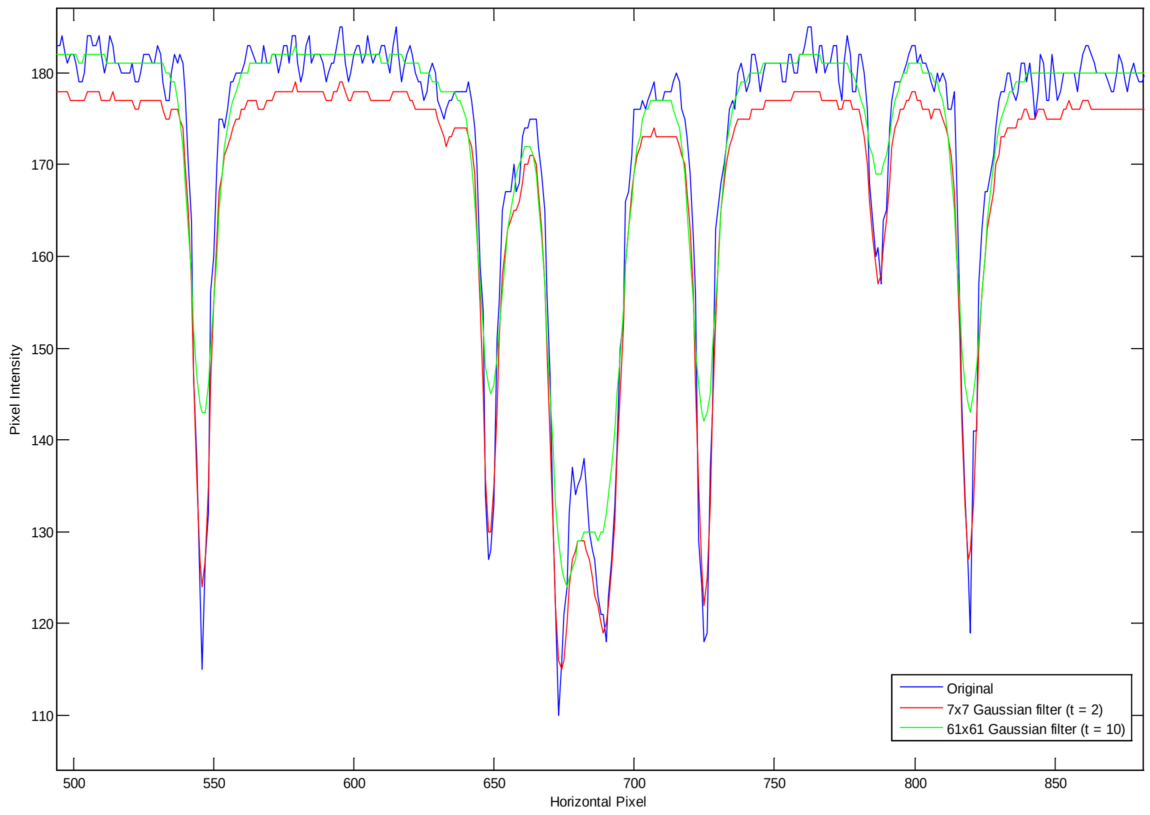

Figure 10. Comparison of line scans (row 406) of red component after Gaussian filters.

The next step after the Gaussian filter is to extract useful information from the image using partial derivatives to find ridges or valleys representing domain boundaries. A ridge or valley is a local maximum or minimum in at least one dimension, which can be visualized as an elongated object at a different intensity level than pixels around it. Ridge and valley detection is commonly used to map out roads or water systems in aerial photographs or to locate blood vessels in medical images. There are several techniques that use partial derivatives that are used to find ridges and valleys. One way to accomplish this is to simply use the partial derivatives along the horizontal and vertical directions (L_x and L_y) of the image. Another option is to calculate partial derivatives aligned with the principal curvatures at each pixel of a height map of the image (intensity is the height). The key is to transform the image coordinate system (x,y) into the p- and q- directions of the principal curvatures by rotating by an angle β defined by Eq. (22) and Eq. (23)[19],

$$\cos\beta = \sqrt{\frac{1}{2}\left(1+\frac{L_{xx}-L_{yy}}{\sqrt{\left(L_{xx}-L_{yy}\right)^2+4L_{xy}{}^2}}\right)} \tag{22}$$

$$\sin\beta = \sqrt{\frac{1}{2}\left(1-\frac{L_{xx}-L_{yy}}{\sqrt{\left(L_{xx}-L_{yy}\right)^2+4L_{xy}{}^2}}\right)}\ . \tag{23}$$

A characteristic of this coordinate system is that $L_{pq} = 0$. The associated directional derivative operators are given by Eq. (24) and Eq. (25),

$$\partial_p = \sin(\beta)\partial_x - \cos(\beta)\partial_y , \tag{24}$$

$$\partial_q = \cos(\beta)\partial_x + \sin(\beta)\partial_y . \tag{25}$$

To detect a ridge or valley with this coordinate system, a set of conditions,

$$\text{Ridge} \begin{cases} L_p = 0 \\ L_{pp} < 0 \\ |L_{pp}| \geq |L_{qq}| \end{cases} \quad \text{or} \quad \text{Valley} \begin{cases} L_q = 0 \\ L_{qq} > 0 \\ |L_{qq}| \geq |L_{pp}| \end{cases} ,$$

The (x,y) and the (p,q) coordinate systems produce nearly equivalent results. While the (p,q) coordinate system was used for image processing of the whole crystal, ridge and valley detection will be demonstrated below for a single image using the (x,y) coordinate system. The procedure is essentially the same.

An additional step of applying a 5x5 wiener filter after calculating each derivative helps minimize noise. A wiener image filter is a low-pass adaptive filter that using statistics from the underlying image to estimate the noise in the image, in order to filter it out. An original section of the image and the images of the derivatives before and after the wiener filter are shown in Figure 12 through Figure 19.

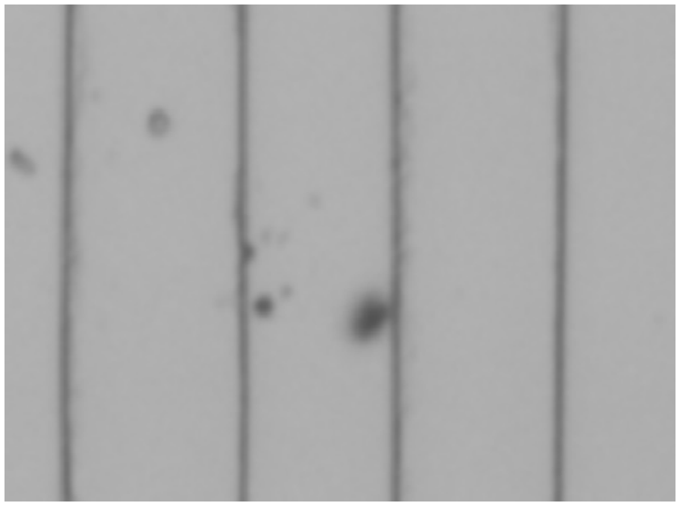

Figure 11. Original section of image.

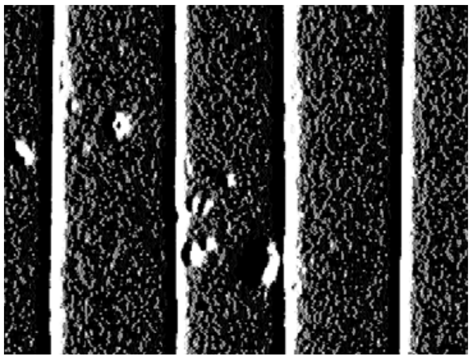
Figure 12. L_x before wiener filter.

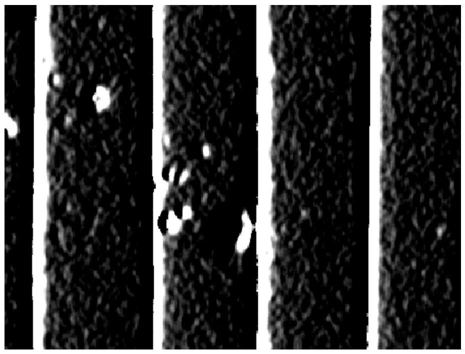
Figure 13. L_x after wiener filter.

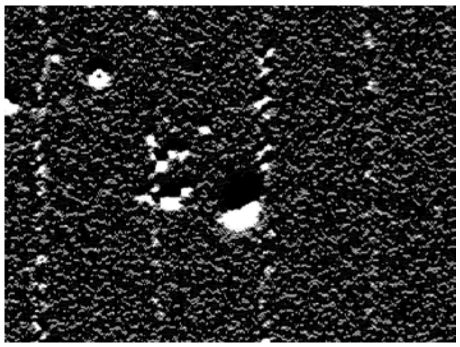
Figure 14. L_y before wiener filter.

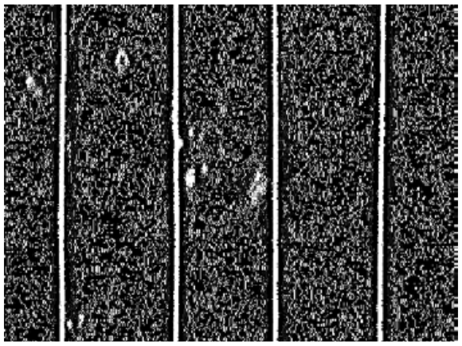
Figure 16. L_{xx} before wiener filter.

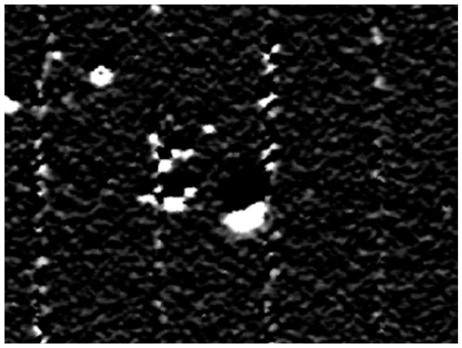
Figure 15. L_y after wiener filter.

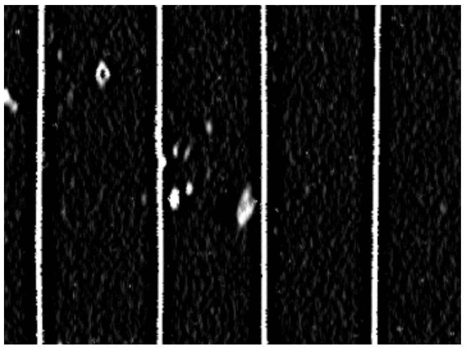
Figure 17. L_{xx} after wiener filter.

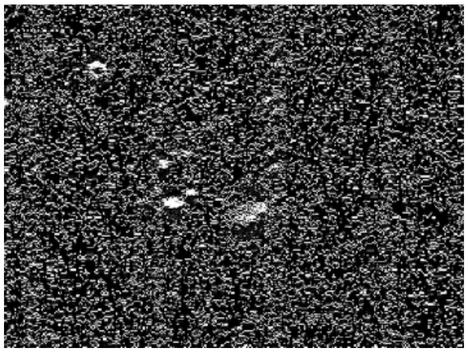
Figure 18. L_{yy} before wiener filter.

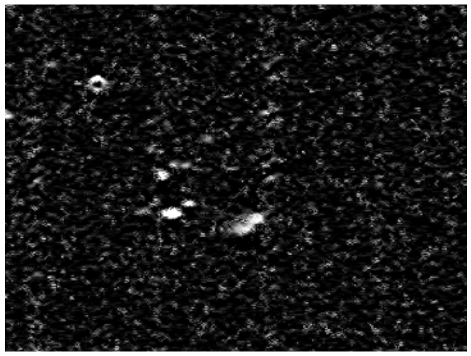
Figure 19. L_{yy} after wiener filter.

Assuming the domain boundaries are mostly vertical, the easiest way to determine the position of the boundary is to scan line by line in the image, looking for the zeros in L_x. To avoid small variations and noise, thresholds for the first and second derivative must be exceeded within a window around a point of interest stretching several pixels. To ignore points that are not on a line, a measure of probability of a blob (such as dirt) was calculated using Eq. (26),

$$Blob = L_{xx} * L_{yy} , \qquad (26)$$

where L_{xx} exceeds a threshold, otherwise the value is 0. An image of the blob value after wiener filtering is shown in Figure 20 for the same region as in Figure 12 through Figure 19.

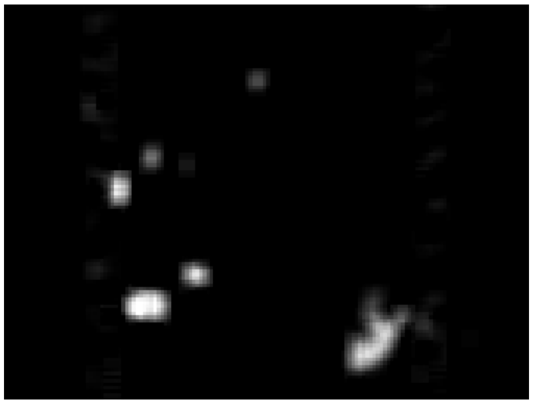

Figure 20. Image of the blob measurement

The potential ridge and valley points overlaid on the original image are shown in Figure 21.

To further reduce potential ridge and valley points to only include points on the domain boundaries, a line detector was constructed. This process determines whether a point appeared to be part of a line. First, a small subsection of the image around the point was compared to the statistical mode of that subsection multiplied by a factor that was manually adjusted for the best results. A threshold image was created and regions of contiguous pixels around the border that exceeded the threshold were counted as long as they were not too small or too large (parameters tuned for the best results). A line would have two regions at the border where the line crossed into and out of the subsection. The line did not have to be straight to satisfy this condition. The only other condition was that the number of pixels above threshold did not exceed a certain percentage of the total pixels in the subsection. Again, the percentage was manually tuned for the best results. This made it unlikely that the pixels in the middle of the threshold image showed anything other than a line. The procedures did not guarantee that the point was part of a line, but it established a good candidate that excluded uncertain features. Usually, only a few points would be removed from the list of potential points, but these were away from the domain boundaries, which can be confusing for the next few processes.

The reduced set of points was next organized into lines by scanning several rows in the middle of the image to determine the approximate number lines. This was accomplished by adding up the points within groups of 3 columns and looking for clusters of points above a threshold. Each cluster was typically separate for each domain boundary. This would not necessary work every time, but in this experiment it did because the boundary lines are straight, vertical, and relatively thin compared to the domain period. The algorithms will need to be more robust as they are further developed.

Next, a point within each cluster was connected to other points by searching up and down within a window of a few pixels to add more points to that line. Here, the assumption is that the boundary points should be connected because they are near to each other. After the lines were established, linear interpolation was used to fill in the gaps.

By filling in regions between the boundary lines with the correct color (black or white), an alternating binary map of the domain structure for a single image is created, as shown in Figure 22. Black was for a negative d coefficient and white was for a positive coefficient. Using row 1000 as an example, the Fourier transform of the binary map is shown in Figure 23, for different levels of zero-padding. Zero-padding is adding extra zeros at the end of the array of numbers to be transformed, which increases the resolution in spatial frequency space. The higher amounts of zero-padding give the most accurate value of the lowest frequency peak in the Fourier transform. The peak value can then be used in the OPG threshold calculation to find the relative pump threshold.

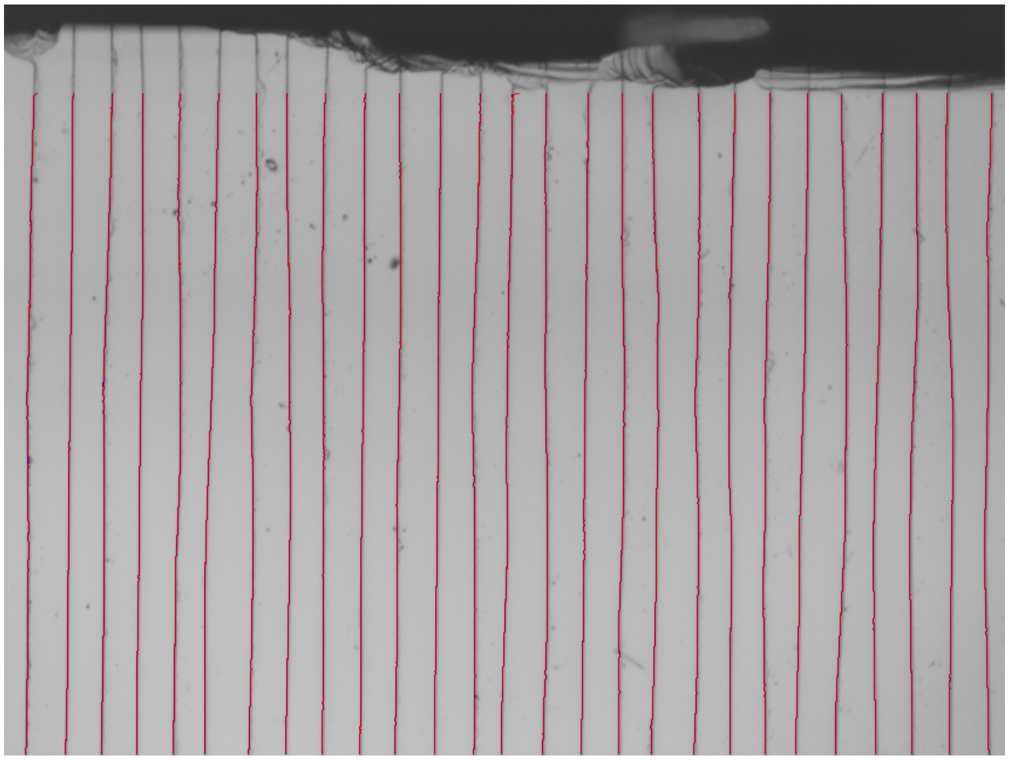

Figure 21. Boundary lines with linear interpolation shown as red dashed line.

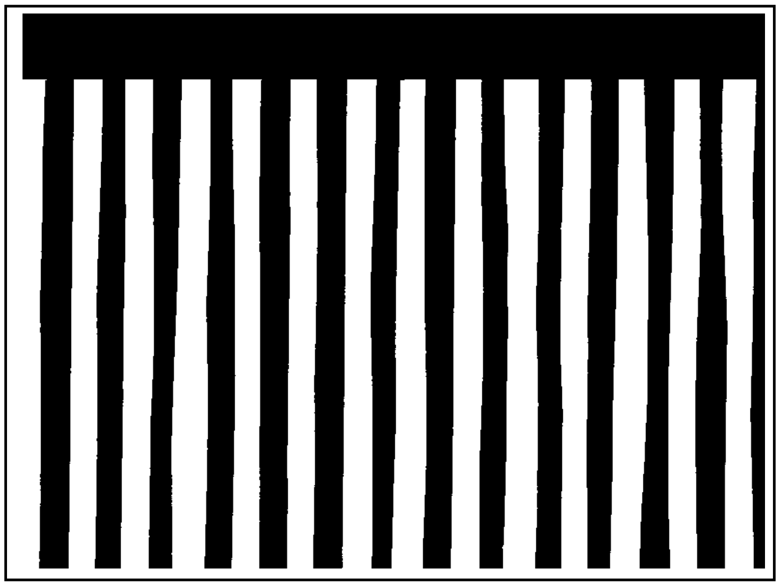

Figure 22. Binary map of domain structure.

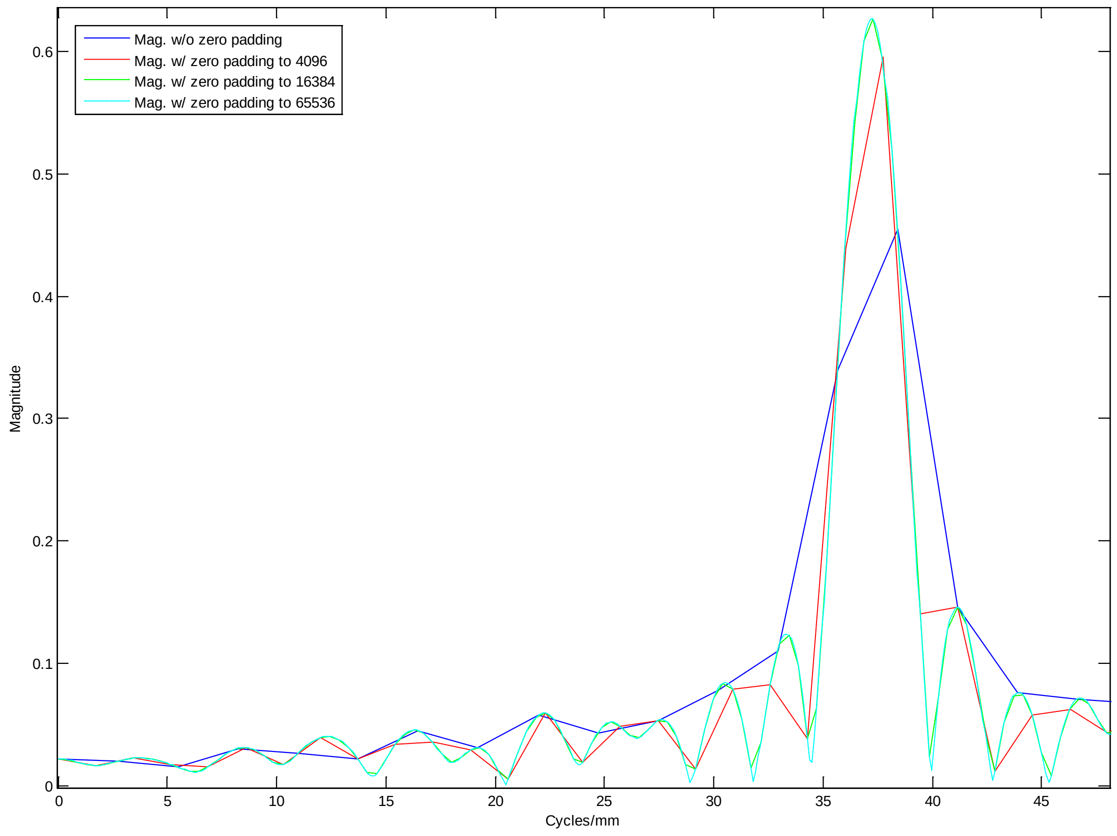

Figure 23. Fourier transform of map at row 1000 using various levels of zero-padding.

When connecting the images together, the complete lines from each image were combined one at a time into a comprehensive list. When each new line was added, it was compared to all of the lines currently in the list to see if it overlapped any by calculating a mean difference between points on each pair of lines. The registration information was added to the points making up the lines first and then they were linearly interpolated to integer pixel locations for direct comparison. If the mean difference quantity was less than a threshold (a value less than 1/6 of the domain period was chosen), then the pair of lines were considered equivalent and the latest line was not included in the comprehensive list. Each row of images was processed in this way until all the domain boundary lines were included. The same process for creating a binary map and calculating the FFT of a single image was then extended to a whole row of images.

There were several thresholds and parameters that need to be determined for these procedures to work correctly. The best way to accomplish this would be to start with a single image and manually adjust the parameters until each process that uses the parameters produces the expected results. The parameters should easily generalize to the whole batch of images, assuming they were all acquired similarly. Eventually, one could conceivably create an algorithm to automatically tune the parameters by knowing the scale and the design of the device, and then comparing the results at each step until they best match what would be expected for a given design.

2.3 OPG Setup

An OPG setup was used to validate the accuracy of the image processing routines. The setup is shown in Figure 24. Where the 1.064μm wavelength laser beam reaches the PPLN crystal, the beam waist ($1/e^2$ diameter) was measured to be approximately 0.4 mm. After aligning the crystal for efficient conversion by ensuring that the back reflections were collinear with the incident beam, the signal and idler were directed to an energy meter for measurement. At this orientation, the Fresnel reflections from the crystal facets lead to a low-finesse monolithic OPO which slightly lowers the threshold as compared to when the crystal is rotated away from the resonant condition. To determine the threshold, the pump energy was adjusted until the combined signal and idler energy reached 1μJ. The pump energy was measured when that condition was met and then the whole process was repeated several times while translating the crystal laterally.

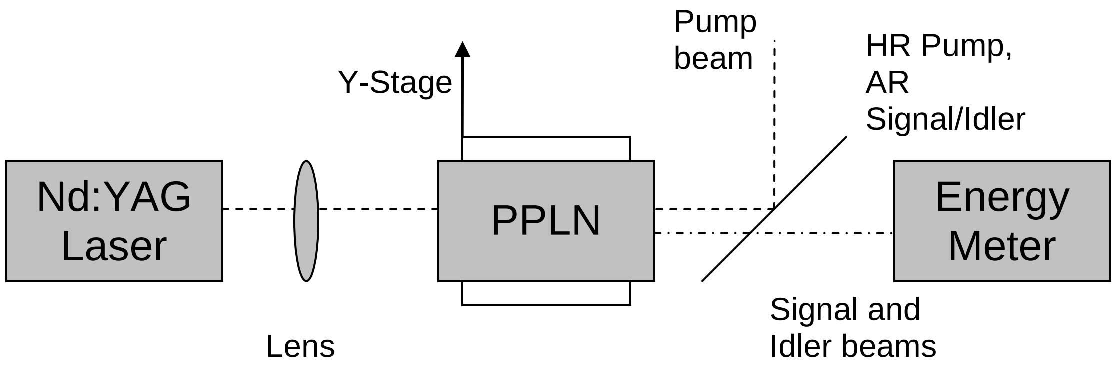

Figure 24. OPG experiment setup.

2.4 Calculated and measured OPG threshold

The crystal's performance was compared to the prediction based on mapping domain boundaries of the entire crystal top surface with the image processing techniques described previously. The measured and calculated pump threshold across the width of the crystal is shown in Figure 25.

Since the calculation was relative, the constant of proportionality was adjusted until the scale of the calculated threshold matched the measured thresholds visually.

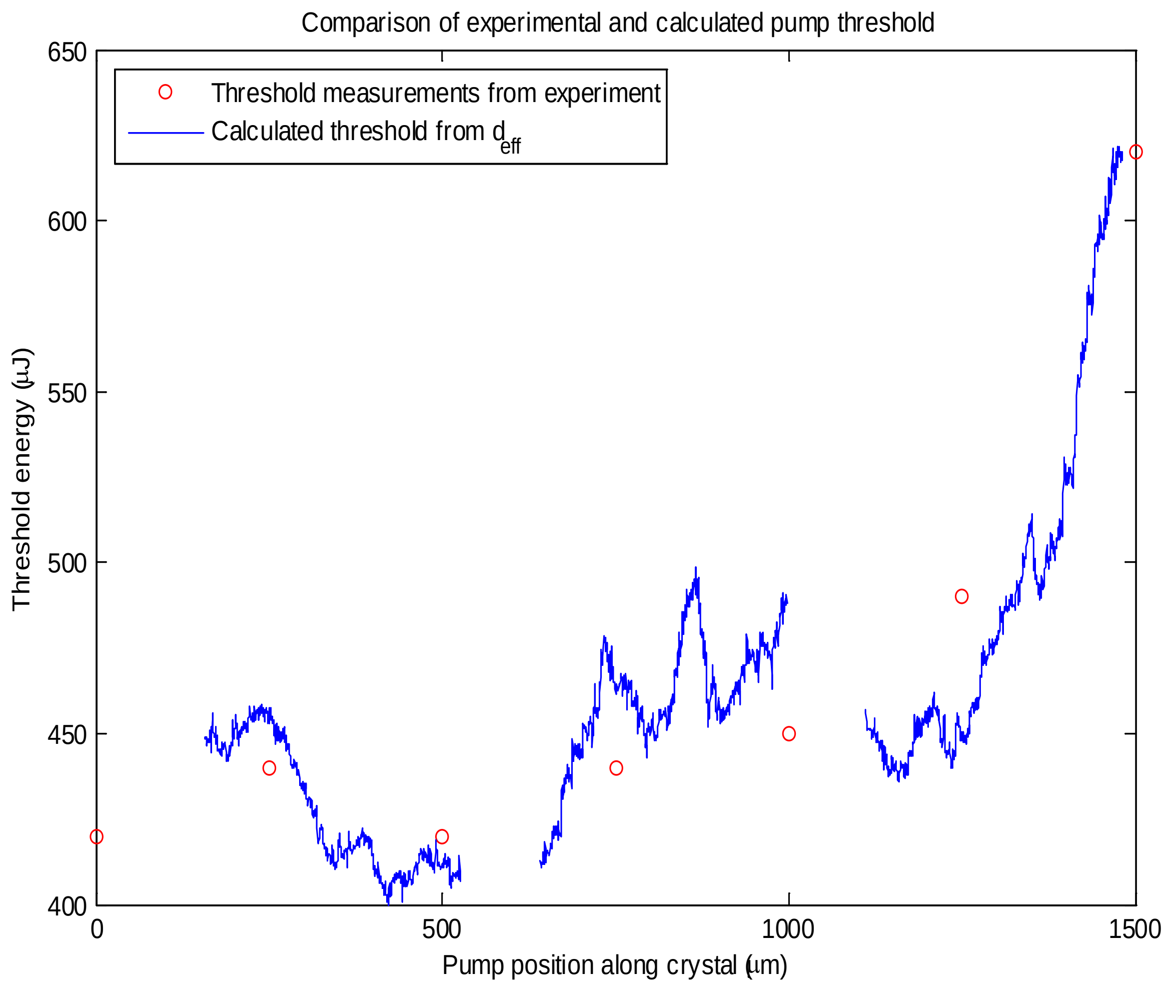

Figure 25. Calculated and measured threshold of PPLN crystal.

For the current experiment, the mosaic of the characterized crystal was comprised of 38 columns and 3 rows of individual microscope images. The columns were registered together, but for this thesis, the rows have not yet been connected, leaving gaps in the calculated threshold data where the domain map is split between rows in the mosaic. The reason for this is that successive images were overlapped significantly in the direction parallel to the gratings, but also had a slight translation perpendicular to this direction, a situation demonstrated in Figure 26. Once the rows are connected together, the calculated threshold data will be continuous across the crystal. The calculations show a result consistent with the performance found with the OPG setup. This important proof of concept shows promise for performing full-scale realistic simulations of quasi-phase matched nonlinear crystals.

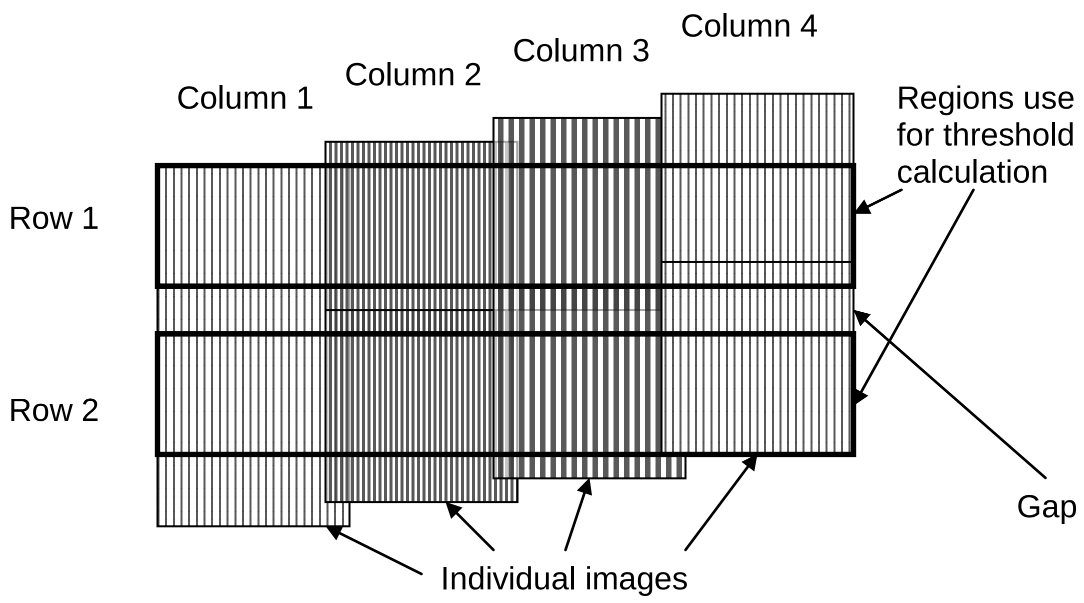

Figure 26. Representative diagram of the mosaic of images.

Chapter 3
Conclusions

A technique for quantitatively evaluating the poling quality of quasi-phase matched crystals has been demonstrated. After stitching a mosaic of microscope images together, the locations of crystal domains were accurately determined through a series of imaging processing steps. The steps include registering the images, Gaussian filtering, using derivates to find ridges or valleys, isolating those potential ridges or valleys into lines, connected the lines, merging the lines from all the images, and then creating a map from these boundary lines. The full crystal map was used to calculate a relative d_{eff} and then subsequently was compared experimentally to an OPG threshold measurement as a function of position in the crystal. The mapping of d_{eff} to threshold matched the OPG measurements, validating this method as a useful means to evaluate nonlinear optical crystal performance. Beyond this proof of concept, the data extracted from the images could easily be applied to other processes like second harmonic generation and be used for any laser wavelength.

There are several avenues for future research. In our case, the crystal was selectively etched periodically poled lithium niobate, but it may be possible to extend this technique to other materials. For example, there should be little problem applying this to lithium tantalate due to its similarity. Also, a properly prepared orientation-patterned GaAs (side polished) would be a possibility. Alternative modes of obtaining the images are also feasible. Under certain viewing conditions, an SEM is capable of producing usable images for mapping domain boundaries. Once the structure has been mapped, realistic nonlinear crystals can be modeled, supporting complex beam propagation simulations.

Appendix: Matlab Code

Listing 1: ImProcess.m

```
% Runs image processing routines.
%
% After collecting the images, each image is first Gaussian
% filtered, and then all the partial derivatives and the
% blob function are calculated. These are used to find the
% potential boundary points, some of which are eliminated because
% they don’t pass the IsLine test. The points are collected into
% lines and then are displayed on the images. All the data for each
% image is saved.

% Initialize parameters
umperpx = 0.14244.*2;   % Scale factor microns/pixel
w = 2;                  % Window for FindBoundary
th = 1;                 % Threshold for 1st deriv.
th2 = 1;                % Threshold for 2nd deriv.
thb = 0.25;             % Threshold for blob
rows = 3;               % Rows in superimage
cols = 38;              % Columns in superimage
% Gets all the filenames for the images
a = dir('*.bmp');
for cnt = 1:size(a,1)
   fn(floor((cnt-1)/cols)+1,mod(cnt-1,cols)+1,:) = a(cnt,1).name;
end

% Iterate through all columns in a row
irow = 1;
for icol = 1:cols
% Clear old variables
clear blob im1 imtest1 Lp Lpp Lq Lqq linedown linep lines lines2 linesz
lineup posx posx2 pxsz px2sz tposx tposx2 tpx2sz tpxsz x y

tic
% Reading in image and applying initial filters
disp(['Reading ' reshape(fn(irow,icol,:),1,16) ' and applying filters']);
im1 = imread(fn(irow,icol,:));  im1 = im1(:,:,2);
[imtest1 Lp Lq Lpp Lqq blob] = Filter(im1);

% Find the potential boundary points
[posx posx2 pxsz px2sz] = FindBoundary(Lp,Lq,Lpp,Lqq,blob,w,th,th2,thb);
disp('Done');
toc

% Gets rid of points that do not pass the IsLine test
disp('Eliminating points that are not lines');
% Ridges first
tposx = [];
tpxsz = zeros(1,size(imtest1,1));
for i = 1:size(imtest1,1)
        for j = 1:pxsz(i)
            if (IsLine(imtest1(max(1,(i-10)):min(size(imtest1,1),(i+10)), ...
                    max(1,(round(posx(i,j))-
10)):min(size(imtest1,2),(round(posx(i,j))+10))),0.9) == 1)
                tpxsz(i) = tpxsz(i) + 1;
                tposx(i,tpxsz(i)) = posx(i,j);
            end
        end
end

% Then valleys
tposx2 = [];
tpx2sz = zeros(1,size(imtest1,1));
for i = 1:size(imtest1,1)
        for j = 1:px2sz(i)
            if (IsLine(imtest1(max(1,(i-10)):min(size(imtest1,1),(i+10)), ...
                    max(1,(round(posx2(i,j))-
10)):min(size(imtest1,2),(round(posx2(i,j))+10))),0.9) == 1)
                tpx2sz(i) = tpx2sz(i) + 1;
                tposx2(i,tpx2sz(i)) = posx2(i,j);
            end
        end
end

disp('Done');
toc

% Converting separate points into connected line
disp('Collecting into domain boundary lines');
[numlines lines linesz] = FindLines(imtest1,tpx2sz,tposx2);
disp('Done');
toc
starty = 1;

% Show filtered image with lines
figure;
imshow(umperpx.*(1:size(im1,2)),umperpx.*(1:size(im1,1)),im1);
xlabel('Distance (\mum)');
ylabel('Distance (\mum)');
hold on;
for i = 1:numlines
    x = lines(i,1:linesz(i),1);
    y = lines(i,1:linesz(i),2);
    % Linear interpolation
    lines2(i,1:(size(im1,1)-starty+1)) = interp1(x,y,starty:size(imtest1,1));
    plot(umperpx.*lines2(i,1:(size(im1,1)-
starty+1)),umperpx.*(starty:size(im1,1)),'r:');
end
toc

% Save all information about image
save(fn(irow,icol,1:12),'lines','lines2','linesz','tposx2','tpx2sz','tposx','
tpxsz');

end
```

Listing 2: Filter.m

```
% Performs image processing filters and derivates
% I = image
% Lpp and Lqq are calculated as eigenvalues of
%     Hessian, [Lxx Lxy; Lxy Lyy], at each point
%
% The image has a Gaussian filter applied, and then
% several partial derivatives (in x and y) are calculated.
% The partial derivatives are then aligned to the principal
% curvatures. Lpp and Lqq are found from the eigenvalues of
% the Hessian matrix. Finally, the blob function is calculated.

function [L Lp Lq Lpp Lqq blob] = Filter(I)

% Perform Gaussian filter and regular partial derivatives
L = GaussFilter(I,3,2);
Lx = imfilter(double(L),[-1/12 2/3 0 -2/3 1/12],'replicate','conv');
Ly = imfilter(double(L),[-1/12; 2/3; 0; -2/3; 1/12],'replicate','conv');
Lxx = imfilter(double(L),[-1/12 4/3 -5/2 4/3 -1/12],'replicate','conv');
Lyy = imfilter(double(L),[-1/12; 4/3; -5/2; 4/3; -1/12],'replicate','conv');
Lxy = imfilter(double(L),[1/4 0 -1/4; 0 0 0; -1/4 0 1/4],'replicate','conv');
% Wiener filter the result
Lx = wiener2(Lx,[5 5]);
Ly = wiener2(Ly,[5 5]);
Lxx = wiener2(Lxx,[5 5]);
Lyy = wiener2(Lyy,[5 5]);
Lxy = wiener2(Lxy,[5 5]);

% Align to principal curvatures
cosb = sqrt(1/2.*(1+(Lxx-Lyy)./sqrt((Lxx-Lyy).^2+4.*Lxy.^2)));
sinb = sqrt(1/2.*(1-(Lxx-Lyy)./sqrt((Lxx-Lyy).^2+4.*Lxy.^2)));
Lp = sinb.*Lx-cosb.*Ly;
Lq = cosb.*Lx+sinb.*Ly;

% Second derivatives found from eigenvalues of given matrix
Lpp = zeros(size(L,1),size(L,2));
Lqq = zeros(size(L,1),size(L,2));
for i = 1:size(L,1)
    for j = 1:size(L,2)
        e = eig([Lxx(i,j) Lxy(i,j); Lxy(i,j) Lyy(i,j)]);
        Lpp(i,j) = e(1);
        Lqq(i,j) = e(2);
    end
end
Lpp = wiener2(Lpp,[5 5]);
Lqq = wiener2(Lqq,[5 5]);
% Calculate blob function, excluding low 2nd derivatives
blob = wiener2((Lpp > 0.2).*(Lqq > 0.2).*(Lpp.*Lqq),[5 5]);
```

Listing 2.1: Gaussian.m

```
% Discrete 2-D Gaussian function
% x = x coordinate
% y = y coordinate
% t = t parameter
function y = Gaussian(x,y,t)
y = 1/(2.*pi.*t).*exp(-(x.^2+y.^2)/(2.*t));
```

Listing 2.2: GaussFilter.m

```
% Applies Gaussian image filter
% I = image
% m = Size of filter on a side divided by 2
% t = t parameter
%
% Creates the specified Gaussian filter and then
% applies it to the given image.
function M = GaussFilter(I,m,t)

% Construct filter
for i = -m:m
    for j = -m:m
        filter1(i+m+1,j+m+1) = Gaussian(i,j,t);
    end
end

% Use it
M = imfilter(I,filter1,'replicate','conv');
```

Listing 3: FindBoundary.m

```
% Finds potential boundary points
% W = window to look for max value
% Th = first derivative threshold
% Th2 = second derivative threshold
% Thb = blob threshold
%
% For each row in the image, the potential ridge and valley
% points are located using the partial derivatives and the
% threshold values. The positions are then compiled into
% a list.

function [pos pos2 psz p2sz] =
FindBoundary(imdp,imdq,imdpp,imdqq,blob,w,th,th2,thb)

% Search one row at a time in the x direction
pos = zeros(size(imdp,1),200);
pos2 = zeros(size(imdp,1),200);
for i = 1:size(imdp,1)
    [p p2] =
FindZeros(imdp(i,:),imdq(i,:),imdpp(i,:),imdqq(i,:),blob(i,:),w,th,th2,thb);
    psz(i) = size(p,2);
    p2sz(i) = size(p2,2);
    if (size(p,1) ~= 0)
        pos(i,1:psz(i)) = p;
    end
    if (size(p2,1) ~= 0)
        pos2(i,1:p2sz(i)) = p2;
    end
end
```

Listing 3.1: Findzeros.m

```
% Finds zero crossing in the first derivatives and matches them
% 2nd derivatives that meet the conditions for a ridge or
% valley
%
% p,q,pp,qq = partial derivatives of one row of image
% w = window to look for max value
% Th = first derivative threshold
% Th2 = second derivative threshold
% Thb = blob threshold
%
% First, the row of pixels is searched from left to right until
% a zero crossing in the first derivative is found. Then the
% thresholds are compared to the second derivatives and the
% blob function. If all of them succeed, the criteria for
% ridges and valleys are applied and the positions that
% satisfy the criteria are saved.

function [pos pos2] = FindZeros(p,q,pp,qq,blob,w,th,th2,thb)

pos = [];
```

```
pos2 = [];
cz = 0;
cz2 = 0;
c = 1;
while (c < size(p,2))
   % Look for a zero crossing
   while ((c < size(p,2)) && (p(c)*p(c+1) > 0) && (q(c)*q(c+1) > 0))
       c = c + 1;
   end
   % When you find one, match it against the critera
   if (c < size(p,2))
       bmax = max(abs(blob(max(1,c-w):min(size(blob,2),c+w))));
       ppmax = max(abs(pp(c:(c+1))));
       qqmax = max(abs(qq(c:(c+1))));
       % Check if it's greater than the threshold values
       if ((bmax < thb) && ((ppmax > th2) || (qqmax > th2)))
           pzero = p(c)*p(c+1);
           qzero = q(c)*q(c+1);
           ppmin = min(pp(c:(c+1)));
           [pmaxl pmaxli] = max(abs(p(max(1,c-w):c)));
           [pmaxr pmaxri] = max(abs(p(c+1:min(size(p,2),c+w))));
           [qmaxl qmaxli] = max(abs(q(max(1,c-w):c)));
           [qmaxr qmaxri] = max(abs(q(c+1:min(size(p,2),c+w))));
           % Check for ridge
           if ((pzero < 0) && (ppmin <= 0) && (ppmax >= qqmax) && (ppmax >
th2) && ...
                   (pmaxl > th) && (pmaxr > th) && (p(max(1,c-w)-
1+pmaxli).*p(c+pmaxri) < 0))
               cz = cz + 1;
               pos(cz) = c-p(c)./(p(c+1)-p(c));
           end
           % Check for valley
           if ((qzero < 0) && (qqmax >= 0) && (qqmax >= ppmax) && (qqmax >
th2) && ...
                   (qmaxl > th) && (qmaxr > th) && (q(max(1,c-w)-
1+qmaxli).*q(c+qmaxri) < 0))
               cz2 = cz2 + 1;
               pos2(cz2) = c-q(c)./(q(c+1)-q(c));
           end
       end
   end
   c = c + 1;
end
```

Listing 4: IsLine.m

```
% Determines if the pixel in the middle of the image subsection
% is likely a line
%
% im = image subsection
% thp = threshold factor for mode of image
%
% First a threshold image is created using a fraction (thp) of
% the statistical mode as a cutoff point. Starting in the upper
% left, the threshold image is searched for sequences above
% threshold around the border starting with the top, then moving
% to the right, bottom, and finally the left. If a sequence
% of at least 4 pixels is found, it is considered a line exiting
% the subsection. After the search is completed, finishing at
% the upper left again, the number of sequences is counted.
% If there are 2 sequences, the sequence lengths are less than 11
% pixels, and the total pixels in the subsection above threshold
% is less than 55% of the subsection, then the middle of the
% subsection is likely to be a part of a line.

function A = IsLine(im,thp)

cnt = 0;
wcnt = 0;
% Create threshold image using thp*mode(im) as divider
th = thp.*mode(mode(double(im)));
last = double(im(1,1)) <= th;
wpos = [];
% Determines line has started in the upper left
if (last)
    wcnt = 1;
    wid(wcnt) = 1;
    wpos = [1; 1];
end
% Start from top border, count the number of the consecutive
% pixels in each line segment around the border and what size
% it is. If it is less than 4 pixels wide, disregard
for i = 2:size(im,2)
    if (last)
        wid(wcnt) = wid(wcnt) + 1;
    end
    if ((double(im(1,i)) <= th) ~= last)
        last = (double(im(1,i)) <= th);
        if (last)
            wcnt = wcnt + 1;
            wid(wcnt) = 1;
            wpos = [1; i];
        else
            if (wid(wcnt) < 4)
                wcnt = wcnt - 1;
            else
                mpos = ([1; i]+wpos)./2;
                mx =
round(linspace(round(size(im,1)./2),mpos(2),size(im,1)));
```

```
                my =
round(linspace(round(size(im,1)./2),mpos(1),size(im,1)));
                for k = 1:size(im,1)
                    mim(k) = im(my(k),mx(k)) < th;
                end
                if (min(mim) ~= 1)
                    wcnt = wcnt - 1;
                end
            end
        end
        cnt = cnt + 1;
    end
end
% Do the same on the right border
for i = 1:size(im,1)
    if (last)
        wid(wcnt) = wid(wcnt) + 1;
    end
    if ((double(im(i,size(im,2))) <= th) ~= last)
        last = (double(im(i,size(im,2))) <= th);
        if (last)
            wcnt = wcnt + 1;
            wid(wcnt) = 1;
            wpos = [i; size(im,2)];
        else
            if (wid(wcnt) < 4)
                wcnt = wcnt - 1;
            else
                mpos = ([i; size(im,2)]+wpos)./2;
                mx =
round(linspace(round(size(im,1)./2),mpos(2),size(im,1)));
                my =
round(linspace(round(size(im,1)./2),mpos(1),size(im,1)));
                for k = 1:size(im,1)
                    mim(k) = im(my(k),mx(k)) < th;
                end
                if (min(mim) ~= 1)
                    wcnt = wcnt - 1;
                end
            end
        end
        cnt = cnt + 1;
    end
end
% And the bottom border
for i = size(im,2):-1:1
    if (last)
        wid(wcnt) = wid(wcnt) + 1;
    end
    if ((double(im(size(im,1),i)) <= th) ~= last)
        last = (double(im(size(im,1),i)) <= th);
        if (last)
            wcnt = wcnt + 1;
            wid(wcnt) = 1;
            wpos = [size(im,1); i];
        else
            if (wid(wcnt) < 4)
```

```
                wcnt = wcnt - 1;
            else
                mpos = ([size(im,1); i]+wpos)./2;
                mx =
round(linspace(round(size(im,1)./2),mpos(2),size(im,1)));
                my =
round(linspace(round(size(im,1)./2),mpos(1),size(im,1)));
                for k = 1:size(im,1)
                    mim(k) = im(my(k),mx(k)) < th;
                end
                if (min(mim) ~= 1)
                    wcnt = wcnt - 1;
                end
            end
        end
        cnt = cnt + 1;
    end
end
% And the left border
for i = size(im,1):-1:1
    if (last)
        wid(wcnt) = wid(wcnt) + 1;
    end
    if ((double(im(i,1)) <= th) ~= last)
        last = (double(im(i,1)) <= th);
        if (last)
            wcnt = wcnt + 1;
            wid(wcnt) = 1;
            wpos = [i; 1];
        else
            if (wid(wcnt) < 4)
                wcnt = wcnt - 1;
            else
                mpos = ([i; 1]+wpos)./2;
                mx =
round(linspace(round(size(im,1)./2),mpos(2),size(im,1)));
                my =
round(linspace(round(size(im,1)./2),mpos(1),size(im,1)));
                for k = 1:size(im,1)
                    mim(k) = im(my(k),mx(k)) < th;
                end
                if (min(mim) ~= 1)
                    wcnt = wcnt - 1;
                end
            end
        end
        cnt = cnt + 1;
    end
end
% Check if it connects back around to the upper left
if ((double(im(1,1)) <= th) ~= last)
    cnt = cnt + 1;
elseif (last)
    wid(1) = wid(1) + wid(wcnt);
    wcnt = wcnt - 1;
end
```

```
% If there are 2 line segments on the border with widths less
% than 11, and the structure does not take up more than 55% of
% the image subsection, then it is probably a line
A = 0;
if ((wcnt == 2) && (max(wid) < 11) && (sum(sum(double(im) <=
th))./(size(im,1).*size(im,2)) <= 0.55))
    A = 1;
end
```

Listing 5: FindLines.m

```
% Constructs lines out of separate boundary points
%
% imtest1 = image
% tpx2sz = size of points making up valley
% tposx2 = valley points
% lines = constructed lines
% linesz = size of each line
% numlines = number of lines
%
% A cluster of every three columns in the image created and the
% number of points within each cluster is counted. Ignoring
% isolated small clusters, the number of lines and the location
% of the clusters is found by looking for groups of clusters with
% points surrounded by clusters with little or no points. The
% locations are then searched for a point within the cluster,
% and these points form the beginning of the lines. Above and
% below the points are searched for other points within
% 8 pixels horizontally and these new points are connected
% to the lines. The next points are searched for within 8
% pixels horizontally of the new points. Once the top and bottom
% of the image are reached, the line is organized to contain the
% points from the top to the bottom of the image.

function [numlines lines linesz] = FindLines(imtest1,tpx2sz,tposx2)

% Looks in bins of three columns for clusters of points
div1 = 3;
pcnt = zeros(1,ceil(size(imtest1,2)./div1) );
for i = (round(size(imtest1,1)./2)-100):(round(size(imtest1,1)./2)+100)
    if (tpx2sz(i) > 0)
        for j = 1:tpx2sz(i)
            pcnt(ceil(tposx2(i,j)./div1)) = pcnt(ceil(tposx2(i,j)./div1)) +
1;
        end
    end
end
% Ignore isolated points
pcnt = pcnt.*(pcnt>7);

% Collect into clusters and mark position
numlines = 0;
i = 1;
while (i < size(pcnt,2))
    while ((pcnt(i) == 0) && (i < size(pcnt,2)))
        i = i + 1;
    end
    if (i < size(pcnt,2))
        numlines = numlines + 1;
        linep(numlines) = i;
    end
    while ((pcnt(i) ~= 0) && (i < size(pcnt,2)))
        i = i + 1;
    end
```

```
end

% Use cluster position to find a starting point for each line
% looking somewhere in the center of the image
lines = zeros(numlines,size(imtest1,1),2);
linesz = ones(1,numlines);
for i = 1:numlines
   potcnt = 0;
   potline = [];
   for j = (round(size(imtest1,1)./2)-100):(round(size(imtest1,1)./2)+100)
       for k = 1:tpx2sz(j)
          if (ceil(tposx2(j,k)./div1) == linep(i))
              potcnt = potcnt + 1;
              potline(potcnt,1:2) = [j; k];
          end
       end
   end
   mcnt = 0;
   for j = 1:potcnt
       mcnt = 0;
       for k = 1:potcnt
           if (abs(tposx2(potline(j,1),potline(j,2))-
tposx2(potline(k,1),potline(k,2))) < 4)
               mcnt = mcnt + 1;
           end
           if (mcnt > 10)
               potstart = j;
               break;
           end
       end
       if (mcnt > 10)
           break;
       end
   end
   % Once you found the point, go and down for point and
   % connect points within 8 pixels of each other
   lineup = [potline(potstart,1);
tposx2(potline(potstart,1),potline(potstart,2))];
   linedown = lineup;
   for j = potline(potstart,1)-1:-1:1
       for k = 1:tpx2sz(j)
           if (tpx2sz(j) > 0)
              if (abs(lineup(2,1)-tposx2(j,k)) < 8)
                  lineup(1:2,1:(size(lineup,2)+1)) = cat(2,[j;
tposx2(j,k)],lineup);
                  break;
              end
              if (abs(mean(lineup(2,1:min(15,size(lineup,2))))-tposx2(j,k)) <
8)
                  lineup(1:2,1:(size(lineup,2)+1)) = cat(2,[j;
tposx2(j,k)],lineup);
                  break;
              end
           end
           if (tposx2(j,k) > lineup(2,1)+2)
              break;
           end
```

```
        end
    end
    for j = (potline(potstart,1)+1):size(tposx2,1)
        for k = 1:tpx2sz(j)
            if (tpx2sz(j) > 0)
               if (abs(linedown(2,size(linedown,2))-tposx2(j,k)) < 8)
                   linedown(1:2,1:(size(linedown,2)+1)) = cat(2,linedown,[j;
tposx2(j,k)]);
                   break;
               end
               if (abs(mean(linedown(2,max(1,(size(linedown,2)-
15)):size(linedown,2)))-tposx2(j,k)) < 8)
                   linedown(1:2,1:(size(linedown,2)+1)) = cat(2,linedown,[j;
tposx2(j,k)]);
                   break;
               end
            end
            if (tposx2(j,k) > linedown(2,size(linedown,2))+2)
                break;
            end
        end
    end
    % Put lines together (top and bottom)
    linesz(i) = size([lineup(1,:) linedown(1,2:size(linedown,2))],2)+2;
    if (lineup(1,1) ~= 1)
        if (linedown(1,size(linedown,2)) ~= size(imtest1))
            lines(i,1:linesz(i),1) = [1 lineup(1,:)
linedown(1,2:size(linedown,2)) size(imtest1,1)];
            lines(i,1:linesz(i),2) = [lineup(2,1) lineup(2,:)
linedown(2,2:size(linedown,2)) linedown(2,size(linedown,2))];
        else
            linesz(i) = linesz(i)-1;
            lines(i,1:linesz(i),1) = [1 lineup(1,:)
linedown(1,2:size(linedown,2))];
            lines(i,1:linesz(i),2) = [lineup(2,1) lineup(2,:)
linedown(2,2:size(linedown,2))];
        end
    else
        if (linedown(1,size(linedown,2)) ~= size(imtest1))
            linesz(i) = linesz(i)-1;
            lines(i,1:linesz(i),1) = [lineup(1,:)
linedown(1,2:size(linedown,2)) size(imtest1,1)];
            lines(i,1:linesz(i),2) = [lineup(2,:)
linedown(2,2:size(linedown,2)) linedown(2,size(linedown,2))];
        else
            linesz(i) = linesz(i)-2;
            lines(i,1:linesz(i),1) = [lineup(1,:)
linedown(1,2:size(linedown,2))];
            lines(i,1:linesz(i),2) = [lineup(2,:)
linedown(2,2:size(linedown,2))];
        end
    end
end
```

Listing 6: MaskStitch.m

```
% Collects the domain boundary lines from all the images
% within a superimage row and puts them into a global
% coordinate system
%
% After gathering the images, the registration information is loaded
% for each image, and put into a translation table, indicating how
% each image needs to be translated to get from the first image to
% the desire image’s location. Then, after cleaning up the line data,
% the line data is translated to the global coordinate system using
% the registration data. The lines are then interpolated to line
% up with integer vertical pixels. Ignore lines touching the border
% of the images, each new line is added to a master list with
% the new coordinate system. As each line is added, up to the previous
% 40 lines are compared to it, and if the mean difference between
% corresponding points is less than 15, then the lines are considered
% the same, and the new line is not added. After all lines are in the
% new list, the range of rows in the global coordinate system that
% the lines are contained within is determined, and then a domain
% map is created. Using the domain map, the FFT of each row is
% calculated and the peak value is saved.

% Initialize parameters
umperpx = 0.14244.*2;
starty = 1;
rows = 3;
cols = 38;
% Get filenames of images
a = dir('*.bmp');
for cnt = 1:size(a,1)
    fn(floor((cnt-1)/cols)+1,mod(cnt-1,cols)+1,:) = a(cnt,1).name;
end

% Load registration data in structure, T, that can be used
load trans
T = zeros(rows,cols,2);
for n = 1:(rows-1)
    for m = 1:(cols-1)
        T(n,m+1,1) = -trans(n,m,1).tdata.T(3,2)+T(n,m,1)-1;
        T(n,m+1,2) = -trans(n,m,1).tdata.T(3,1)+T(n,m,2)-1;
    end
    if (n ~= rows-1)
        T(n+1,1,1) = -trans(n,1,2).tdata.T(3,2)+T(n,1,1)-1;
        T(n+1,1,2) = -trans(n,1,2).tdata.T(3,1)+T(n,1,2)-1;
    end
end
for m = 1:(cols-1)
    T(rows,m,1) = -trans(n,m,2).tdata.T(3,2)+T(rows-1,m,1)-1;
    T(rows,m,2) = -trans(n,m,2).tdata.T(3,1)+T(rows-1,m,2)-1;
end
T(rows,cols,1) = -Tf.tdata.T(3,2)+T(rows,cols-1,1)-1;
T(rows,cols,2) = -Tf.tdata.T(3,1)+T(rows,cols-1,2)-1;

% For a row
```

```
for mrow = 3:3
    % Mark top and bottom of rows
    Trow = 1+max(T(1,:,1));
    Brow = size(im1,1)+min(T(rows,:,1));
    Tl2 = zeros(1835,Brow);
    Tl2s = 0;
    dsz = zeros(rows,cols);
    for n = mrow:mrow;
        for m = 1:cols
            % Get information on image(n,m)
            load(fn(n,m,1:12));
            numlines = size(lines,1);
            % Eliminate extra points (multiple points at same vertical
position
            for i = 1:numlines
                if (sum((diff(lines(i,:,1)) <= 0)-(lines(i,2:size(lines,2),1)
== 0)))
                    f = find(diff(lines(i,:,1)) <= 0);
                    lines(i,(f(1)+1):size(lines,2),1) =
zeros(1,size(lines,2)-(f(1)+1)+1);
                end
                linesz(i) = min(find(lines(i,:,1) == 0))-1;
            end
            % Transfer local (image) line data to global (superimage)
            % line data
            for i = 1:numlines
                x = lines(i,1:linesz(i),1);
                y = lines(i,1:linesz(i),2);
                ls = linesz(i);
                % Ignore lines on right or left image border in
                % case they extend between images
                if ((min(y) >= 2) && (max(y) <= size(im1,2)-2))
                    lines2(i,1:(size(im1,1)-starty+1)) =
interp1(x,y,starty:size(im1,1));
                    l =
interp1((1+T(n,m,1)):(size(im1,1)+T(n,m,1)),lines2(i,:)+T(n,m,2),...

(ceil(1+T(n,m,1))):floor(size(im1,1)+T(n,m,1)),'linear');
                    skip = false;
                    % Look at up to the last 40 lines added and compare
                    % to new line, if match (mean difference less 15 pixels
                    % away) then ignore new line since it was already
                    % included from a different image
                    for q = max(1,(Tl2s-40)):Tl2s
                        d = (l((max(ceil(1+T(n,m,1)),Trange(q,1))-
ceil(1+T(n,m,1))+1): ...
                            (min(floor(size(im1,1)+T(n,m,1)),Trange(q,2))-
ceil(1+T(n,m,1))+1)) ...
                            -
Tl2(q,max(ceil(1+T(n,m,1)),Trange(q,1)):min(floor(size(im1,1)+T(n,m,1)),Trang
e(q,2))));
                        if (abs(mean(d)) <= 15)
                            dm(n,m,dsz(n,m)+1) = mean(d);
                            ds(n,m,dsz(n,m)+1) = std(d);
                            dsz(n,m) = dsz(n,m) + 1;
                            skip = true;
                            break;
                        end
                    end
                    % If not already added, include in master list of lines
                    if (~skip)

Tl2((Tl2s+1),(ceil(1+T(n,m,1))):floor(size(im1,1)+T(n,m,1))) = ...
                            l;
                        Tl2s = Tl2s + 1;
                        Trange(Tl2s,:) = [ceil(1+T(n,m,1));
floor(size(im1,1)+T(n,m,1))];
                    end
                end
            end
        end
    end
    % Determine what rows contain the lines and display all the lines
    figure;
    for k = 1:Tl2s
        hold on;
        a = 1;
        b = Brow;
        for j = 1:Brow
            if (Tl2(k,j) ~= 0)
                a = j;
                break;
            end
        end
        for j = (a+1):Brow
            if (Tl2(k,j) == 0)
                b = j-1;
                break;
            end
        end
        plot(Tl2(k,a:b),a:b,'b');
    end

    % Construct domain map and record FFT peaks for each row
    starty = 3900;
    endy = 5250;
    % starty = 2250;
    % endy = 3500;
    % starty = 550;
    % endy = 1850;
    looky = 1300;
    figure;
    for cy = 1:50:looky
        mask = -1.*ones(50,round(endx+T(1,cols,2)));
        for i = 1:50
            pos2 = [startx Tl2(:,i+starty+cy-1)' endx+T(1,cols,2)];
            for j = 1:2:Tl2s+1
                mask(i,round(pos2(j)):round(pos2(j+1))) =
ones(1,round(pos2(j+1))-round(pos2(j))+1);
            end
        end
        mask(1:50,1:startx) = zeros(50,startx);
        mask(1:50,(endx+T(1,cols,2)+1):2.^17) = zeros(50,2.^17-
(endx+T(1,cols,2)));
```

```
        imshow(mask);

image(umperpx.*(1:size(mask,2)),umperpx.*(1:size(mask,1)),127.*(mask+1));
        xlabel('Distance (\mum)');
        ylabel('Distance (\mum)');

        for i = 1:50
            fft1 = abs(fft(mask(i,:)',2.^17)./size(mask,2));
            [a b] = max(fft1(100:2.^17-100));
            pk(mrow,i+cy-1) = a;
        end
    end

end

% Compare FFT peaks with ideal 1st order QPM
figure;
plot(1:size(pk,2),pk(3,1:size(pk,2))./(2./pi));
xlabel('Vertical position (pixel)');
ylabel('d (measured)/d_1');
title('Actual d_e_f_f performance compared to ideal 1st order QPM');
```

Listing 7: Register.m

```
% Registers each image to the images to the right and below
% then saves the translation relationships
%
% startx = global horizontal position at beginning of crystal
% endx = global horizontal position at end of crystal
%
% After gathering all of the image filenames, each image except
% in the last row or in the last column are loaded along with
% its neighbor to the right and below. A point on the image
% and the corresponding point on each of its neighbors is
% allowed to be selected. These points are registered to each
% other using a cross correlation to improve precision. The
% image in the last row, but second last column is then
% registered to the image in the last row and column. Then
% a point on the first image is used to show the horizontal
% position of the beginning of the crystal and the same with
% the last column in the first row for the end of the crystal.

% Gathers filenames for each image
rows = 3;
cols = 38;
a = dir('*.bmp');
for cnt = 1:size(a,1)
   fn(floor((cnt-1)/cols)+1,mod(cnt-1,cols)+1,:) = a(cnt,1).name;
end

% Load each image and register it to the image to the right and below
warning off;
colors = 2;
overlap = 0.15;
im1 = imread(fn(1,1,:));
im1 = im1(:,:,2);
for rcnt = 1:(rows-1)
   for ccnt = 1:(cols-1)
      % Open right and below images
      im2 = imread(fn(rcnt,ccnt+1,:));
      im3 = imread(fn(rcnt+1,ccnt,:));
      im2 = im2(:,:,2);
      im3 = im3(:,:,2);
      % Allow user to select matching points in right image
      [im1pt,im2pt] = cpselect(im1,im2,'Wait',true);
      % Use correlation to improve registration precision
      im1ptcr = cpcorr(im1pt,im2pt,im1,im2);
      % Save data
      pt12(rcnt,ccnt,1,1:size(im1ptcr,1),1:2) = im1ptcr;
      pt12(rcnt,ccnt,2,1:size(im2pt,1),1:2) = im2pt;
      % Convert data point relation into transform
      t12 = cp2tform(im1ptcr,im2pt,'linear conformal');
      trans(rcnt,ccnt,1) = t12;
      % Do again for below image
      [im1pt2,im3pt] = cpselect(im1,im3,'Wait',true);
      im1pt2cr = cpcorr(im1pt2,im3pt,im1,im3);
      pt13(rcnt,ccnt,1,1:size(im1pt2cr,1),1:2) = im1pt2cr;
      pt13(rcnt,ccnt,2,1:size(im3pt,1),1:2) = im3pt;
      t13 = cp2tform(im1pt2cr,im3pt,'linear conformal');
      trans(rcnt,ccnt,2) = t13;
      trans(rcnt,ccnt,1).tdata.T
      trans(rcnt,ccnt,2).tdata.T

      im1 = im2;
   end
   % Load first image in next superimage row
   im1 = imread(fn(rcnt+1,1,:));
   im1 = im1(:,:,2);
end

% Register bottom right image with image to left
im1 = imread(fn(rows,cols-1,:));
im1 = im1(:,:,2);
im2 = imread(fn(rows,cols,:));
im2 = im2(:,:,2);
[im1pt,im2pt] = cpselect(im1,im2,'Wait',true);
im1ptcr = cpcorr(im1pt,im2pt,im1,im2);
Tf = cp2tform(im1ptcr,im2pt,'linear conformal');

% Allow user to select beginning of crystal
im1 = imread(fn(1,1,:));
im1 = im1(:,:,2);
figure; imshow(im1);
[x y] = ginput(1);
startx = x;

% and end of crystal
im1 = imread(fn(1,cols,:));
im1 = im1(:,:,2);
figure; imshow(im1);
[x y] = ginput(1);
endx = x;

save Trans pt12 pt13 trans Tf startx endx
```

Listing 8: Review.m

```
% Checks to make sure that there are no duplicate domain boundary
% lines within the master list. If there is, they are removed
%
% im1 = image
% numlines = number of lines in image
% lines = list of lines (collection of data points)
% lines2 = list of interpolated lines (horizontal position
%            for each vertical pixel in line)
%
% For each line, the next line is compared point by point. Each point
% within 1 pixel of the next line’s corresponding point is counted.
% If 90% of the points match by this criteria, the lines are
% considered the same and one of them is eliminated.

function [numlines lines linesz lines2] =
Review(im1,numlines,lines,linesz,lines2);

% Search through each line
for i = 1:(numlines-1)
    if (i > numlines-1)
        break;
    end
    matches = 0;
    % Check point in line for a match in another line
   for j = 1:size(im1,1)
       if (abs(lines2(i,j)-lines2(i+1,j)) <= 1)
           matches = matches + 1;
       end
   end
   % Try to fix mistake (eliminate duplicates)
   if (matches > 0.9*size(im1,1))
       if (i < numlines-1)
           linesz = [linesz(1:i) linesz(i+2:numlines)];
           lines =
cat(1,lines(1:i,1:size(lines,2),1:2),lines((i+2):numlines,1:size(lines,2),1:2
));
           lines2 =
cat(1,lines2(1:i,1:size(lines,2)),lines2((i+2):numlines,1:size(lines,2)));
           numlines = numlines - 1;
       else
           linesz = [linesz(1:i) linesz(i+2:numlines)];
           lines = lines(1:i,1:size(lines,2),1:2);
           lines2 = lines2(1:i,1:size(lines,2));
           numlines = numlines - 1;
       end
   end
end
```

References

1. S. M. Russell, P. E. Powers, M. J. Missey, and K. L. Schepler, “Broadband mid-infrared generation with two dimensional quasi-phase-matched structures,” IEEE J. Quantum Electron. QE-37, 877-887 (2001).

2 . A. Norton, and C. de Sterke, “Aperiodic 1-dimensional structures for quasi-phase matching,” Opt. Exp. 12(5), 841-846 (2004).

3 . M. M. Fejer, G. A. Magel, D. H. Jundt, and R. L. Byer, “Quasi-phase-matched second harmonic generation: tuning and tolerances,” IEEE J. Quantum Electron. QE-28, 2631-2654 (1992).

4. R. Boyd, “Nonlinear Optics,” 2nd ed, Academic Press, San Diego, 2003.

5 . R. L. Sutherland, “Handbook of Nonlinear Optics,” Marcel Dekker, Inc. New York, 115-116 (1996).

6 . D. H. Hum, and M. M. Fejer, “Quasi-phasematching,” C. R. Physique 8 (2007).

7 . J.A. Armstrong, N. Bloembergen, J. Ducuing, and P.S. Pershan, “Interactions between light waves in a nonlinear dielectric.” Physical Rev. 127:1918-1925 (1962).

8 . J.D. McMullen, “Optical parametric interactions in isotropic materials using a phase-corrected stack of nonlinear dielectric plates.” J. Appl. Phys. 46, 3076 (1975).

9 . M. Yamada, N. Nada, M. Saitoh, and K. Watanbe, “First order quasi-phase matched LiNbO3 waveguide periodically poled by applying an external field for efficient blue second-harmonic generation.” Appl. Phys. Lett., 62:435-436 (1993).

10 . L.E. Myers, R.C. Eckardt, M.M. Fejer, and R.L. Byer, “Quasi-phase-matched optical parametric oscillators in bulk periodically poled LiNbO3” Opt. Lett. 20:52-54 (1995).

11 . E. Soergel, “Visualization of ferroelectric domains in bulk single crystals,” Appl. Phys. B 81, 729-752 (2005).

12 . S. Zhu, and W. Cao, “Imaging of 180o Ferroelectric Domains in LiTaO3 by Means of SEM,” Phys. Stat. Sol. (a) 173, 495 (1999).

13. G. Rosenman, A. Skliar, I. Lareah, N. Angert, M. Zeitlin, and M. Roth, “Observation of ferroelectric domain structures by secondary-electron microscopy in as-grown KTiOPO4 crystals,” Phys. Rev. B 54, 6222 (1996).

14 . K. Daneshvar, E. A. Giess, A. M. Bacon, D. G. Dawes, L. A. Gea, and L. A. Boatner, “Ion exchange in potassium titanyl phosphate,” Appl. Phys. Lett. 71(6), 756-758 (1997).

15 . C. L. Sones, S. Mailis, W. S. Brocklesby, R. W. Eason, and J. R. Owen, “Differential etch rates in z-cut LiNbO3 for variable HF/HNO3 concentrations,” J. Mater. Chem. 12, 295-298 (2002).

16 . F. Laurell, M. G. Roefols, M. Bindloss, H. Hsuing, A. Suna, and J. D. Beirlein, “Detection of ferroelectric domain reversal in KTiOPO4 waveguides,” J. Appl. Phys. 71, 4664 (1992).

17 . B. Zitova, and J. Flusser, “Image registration methods: a survey,” Image and Vision Computing 21, 977-1000 (2003).

18 . A. P. Witkin, “Scale-space filtering", Proc. 8th Int. Joint Conf. Art. Intell., 1019-1022 (1983).

19 . T. Lindeberg, “Edge detection and ridge detection with automatic scale selection,” Int. J. of Computer Vision 30(2) 1-46 (1998).